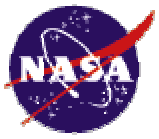
NASA/CR—2000-209914



ICEG2D - An Integrated Software Package for Automated Prediction of Flow Fields for Single-Element Airfoils with Ice Accretion

David S. Thompson and Bharat K. Soni
Mississippi State University, Mississippi State, Mississippi

February 2000

Abstract

An integrated software package, ICEG2D, was developed to automate computational fluid dynamics (CFD) simulations for single-element airfoils with ice accretion. ICEG2D is designed to automatically perform three primary functions: 1) generating a grid-ready, surface definition based on the geometrical characteristics of the iced airfoil surface, 2) generating a high-quality grid using the generated surface point distribution, and 3) generating the input and restart files needed to run the general purpose CFD solver NPARC. ICEG2D can be executed in batch mode using a script file or in an interactive mode by entering directives from a command line. This report summarizes activities completed in the first year of a three-year research and development program to address issues related to CFD simulations for aircraft components with ice accretion. Specifically, this document describes the technology employed in the software, the installation procedure, and a description of the operation of the software package. Validation of the geometry and grid generation modules of ICEG2D is also discussed.

Table of Contents

List of Figures

List of Tables

1. Introduction

Prediction of ice accretion on a full aircraft configuration and the resulting performance degradation is a long-term objective of icing research. To this end, a three-year research and development program has been initiated to address issues related to computational fluid dynamics simulations for aircraft components with ice accretion. The first-year objective of the effort was to develop and validate a software package to perform the preprocessing necessary to generate NPARC [1] solutions automatically for single-element airfoil geometries with ice accretion. It should be noted that the "first year" of the grant actually was nine months in length (March 1 - November 30, 1999). Below is a brief summary of the activities accomplished to date:

- The software framework was developed along with the driver program, ICEG2D, which serves as the interface to the framework and the various component modules. The interface was written in Perl and uses a vocabulary of directives to associate external data with the execution of appropriate program modules. ICEG2D can be executed in batch mode using the directives as a scripting language or in an interactive mode from a command line prompt.

- The geometry module, ICE_DIST_2D, was developed to generate a discrete surface definition for a given iced airfoil configuration. ICE_DIST_2D was written in C with a FORTRAN90 namelist routine and utilizes NURBS routines from GGLib - a geometry/grid library developed at MSU [2]. Automatic distribution of points on the airfoil surface is accomplished using an equidistribution algorithm in the iced region (with the weight function based on surface curvature) and algebraic stretching for the remainder of the airfoil.

- The grid module, ICE_GRID_2D, was developed to generate a nearly orthogonal grid from a previously defined surface distribution. ICE_GRID_2D was written in FORTRAN90 with a C driver routine and uses a parabolic grid generation scheme [3-7]. Both single- and double-block grid topologies can be generated automatically.

- The modules to generate the NPARC namelist input file and NPARC restart file, NPARC_INPUT and NPARC_RESTART respectively, were developed. Both modules were written in FORTRAN90 with C driver routines.

- A preliminary validation of the ICEG2D software package has been performed. Included in the validation is an assessment of grid quality by comparison of NPARC solutions computed using the automatically generated grids with experimental data (NASA GRC data for NLF0414 [8]).

In the sections that follow, the technology employed in ICEG2D, the installation procedure, and a description of the operation of the software package are discussed. Procedures used to validate the software package are also discussed.

2. The ICEG2D Software Package

2.1 General Overview

The ICEG2D software package was developed to automatically perform the preprocessing tasks necessary to generate computational fluid dynamic simulations of flow fields around single-element airfoils with ice accretion using the structured flow solver NPARC [1]. ICEG2D includes a framework in which various operations are performed to generate the CFD solutions. The user interface to the ICEG2D framework is a Perl program that utilizes a system of file associations and directives to perform the five basic functions shown in Table 1.

Function	Module	Details
Generate discrete definition of airfoil surface	ICE_DIST_2D	The surface definition module takes the iced airfoil definition, the clean airfoil definition, and default or user-specified parameters to generate the grid ready, point distribution defining the iced airfoil.
Generate grid	ICE_GRID_2D	The grid generation module takes the iced airfoil surface definition and default or user-specified parameters and blocking specification to generate a nearly orthogonal grid.
Generate NPARC input file	NPARC_INPUT	This module generates an NPARC namelist input file using the generated grid and a user-defined namelist file.
Generate NPARC restart file	NPARC_RESTART	This module generates an NPARC restart file using the generated grid and a user-defined namelist file.
Generate CFD solution		A batch process can be started to run an NPARC solution using the input and restart files.

Table 1. Functions of ICEG2D Interface

There are also directives that execute external procedures to perform auxiliary tasks. For example, FAST [9] can be used to plot the surface distribution or the grid using “display distribution” or “display grid” respectively. A listing of the ICEG2D directives and a brief description of each directive are included in Section 2.4.

Based on discussions with NASA GRC personnel and within the context of automation, it was decided to minimize the user input needed to run ICEG2D. ICEG2D may be executed in batch or interactive mode. In batch mode, a text file containing a stream of ICEG2D directives is used as input via redirection. In the interactive mode, the user executes ICEG2D and issues directives from a command line within a Unix shell. Further enhancing the automation of ICEG2D are the FORTRAN namelists that define the various parameters used to generate the surface distribution and the grid. The namelist variables have been assigned default values that should work for a wide variety of iced airfoil applications.

A typical ICEG2D session consists of taking an iced airfoil definition, available from LEWICE [10] or IRT data [11], and a clean airfoil definition as input to the module ICE_DIST_2D to generate a discrete surface distribution. The surface definition is then used as initial data for the grid generation module, ICE_GRID_2D, to generate a grid. The grid is used along with a problem definition namelist file as input to appropriate modules to generate an NPARC restart file and an NPARC input file. Finally, an NPARC solution is started as a background process. It should be noted that the process could be initiated at any point provided the necessary data files are available and have been defined appropriately within the ICEG2D framework.

ICEG2D was designed with modularity in mind. Module parts or entire modules can be easily replaced without requiring significant changes to the framework.

2.2 Installation

Installation of ICEG2D can be accomplished by following these steps:

Unzip the gzipped file using “gunzip ICEG2D.tar.gz”
Untar the file using “tar -xvf ICEG2D.tar”
Execute “Install_ICEG2D”

The executable “Install_ICEG2D” compiles and links the various modules in their respective directories. Note: Warnings will occur as the NURBS library is being compiled. This is normal and does not affect execution of the code.

During installation, nine subdirectories are created in the main ICEG2D directory. The directories and their contents are given below:

1. examples – four subdirectories containing example cases
2. geometry - ICE_DIST_2D module
3. grid_gen - ICE_GRID_2D modules
4. help_files - help file
5. nparc_gen - two subdirectories for NPARC_INPUT and NPARC_RESTART modules
6. nurbs_lib - GGLib
7. parameter_files - FORTRAN namelist files
8. script_files - contains script files for display directive
9. report – contains this document as a pdf file

These directory names must be left unaltered because of internal file associations made within the ICEG2D framework.

To execute ICEG2D, the following four lines must be included in your .cshrc file:

setenv ICEG2D_DIR (path to directory where ICEG2D is located)

```
setenv NPARC_DIR (path to directory where nparc2ds is located)
set path = ($path $ICEG2D_DIR)
setenv TRAP_FPE "ALL=COUNT;UNDERFL=ZERO;
        OVERFL=IEEE,TRACE(5),ABORT(100); DIVZERO=ABORT;
        INVALID=TRACE(1),ABORT(1)"
```

The first two lines define environment variables that specify the main ICEG2D directory and the directory where the NPARC executable, nparc2ds, is located. The third line includes the ICEG2D directory in the shell $path variable. The fourth line defines how floating point exceptions are treated and is important for error trapping in the fully automatic mode.

2.3 The ICEG2D Framework

The ICEG2D framework is based on a system of internal associations with default or user-prescribed file names. The primary association is the *case_name*. By defining the association *case_name*, a set of automatically defined file names is utilized for input and output to and from the modules. These file names are defined using *case_name* and a default extension based on the file type. Below is a table containing the default file extensions and their associated file types used in the ICEG2D framework.

Extension	**Data**	**Format**
.cln	Clean airfoil input	Number of points, Real (x,y) data pairs
.dat	Iced airfoil input	Number of points, Real (x,y) data pairs
.dst	Airfoil point distribution	Plot3d formatted data, multizone, 2d
.grd	Grid file	Plot3d unformatted data, multizone, 2d
.inp	NPARC input file	FORTRAN namelist file
.rst	NPARC restart file	FORTRAN unformatted file
.out	NPARC standard output	ASCII

Table 2. Default File Extensions

Additionally, there are nine FORTRAN namelists used to define parameters used by ICEG2D modules. The default namelist files are located in the directory $ICEG2D_DIR/parameter_files:

The file containing data for the FORTRAN namelist DISTRIBUTION (associated with *dist_parm_file*) is either **dist_smooth.ini** or **dist_rough.ini** based on the *surface_type* association or a user-defined file. These parameters control the distribution of points on the iced airfoil surface.

The file for namelists NUMBER_OF_BLOCKS and BLOCK_PARAMETERS (associated with *block_parm_file*) is either **single_block.ini** or **double_block.ini** or a user defined file. These parameters define the blocking scheme.

The file for namelist GRID_PARAMETERS (associated with *grid_parm_file*) is **grid_gen.ini** or a user defined file. These parameters control the grid generation process.

The namelists used in the NPARC problem definition - TITLES, INPUTS, TURBIN, SEQDT, and BLOCK - are contained in the problem definition file **NPARC_Info.npc** (associated with *nparc_info_file*) or a user defined file. Typically, the user will copy this file to the local directory and make modifications appropriate for the problem.

Again, these associations may be changed using appropriate ICEG2D directives. Minimally, the user must provide the iced and clean airfoil input files and the problem definition file. **Default values for all parameters are defined within ICEG2D**. These default values were determined by trial and error and have proven effective for the cases examined to date. In some cases, the user may desire to modify one or more of the namelist variables. The suggested procedure is to copy the needed files into the directory containing the airfoil definition files and make the necessary modifications. The modified file can then be associated with the internal file using the appropriate ICEG2D directive.

Other files are generated during the course of the NPARC solution. The user is referred to the NPARC manual [1] for a description of these files.

2.4 The ICEG2D Interface

The ICEG2D framework provides a series of associations between internal data and external files. Some associations are created automatically while others are defined using the ICEG2G directives. The ICEG2D directives all have the same form:

[*directive – action*] [*object - what the directive produces/acts on*] [*additional info (if needed)*]

Each directive **must** start in the first column, i.e., there must be no leading blanks. Either upper or lower case letters must be used for the directives and the objects. File names are case sensitive. In the descriptions below, the directives and objects appear in bold while the data entries appear in Italics. Also, characters appearing in parentheses are optional.

1. # - Indicates that a comment follows. A # is needed for each line of the comment

2. **def(ine)** – defines something
   - **mode** *mode_type* [**int(eractive)** or **bat(ch)** with interactive being the default. The program will exit if an error is encountered in batch mode]
   - **surface** *surface_type* [**smooth** or **rough** with **smooth** being the default. Sets the default distribution parameter file to **dist_smooth.ini** or **dist_rough.ini** respectively.]
   - **case** *case_name* [identifier used for default associations]
   - **iced_geom** *iced_geom_file* [iced geometry file name, default is *case_name.dat*]

- **clean_geom** *clean_geom_file* [clean geometry file name, default is *case_name.cln*]
- **dist** *dist_file* [distribution file name, default is *case_name.dst*]
- **dist_parm** *dist_parm_file* [file containing distribution parameters or "default" to use the default distribution parameter file for the specified *surface_type*]
- **grid** *grid_file* [grid file name, default is *case_name.grd*]
- **grid_parm** *grid_parm_file* [file containing grid parameters or "default" to use default grid parameter file **grid_gen.ini**]
- **block_parm** *grid_block_file* [file containing blocking parameters, "single" for default single-block definition using **single_block.ini**, "double" for default double-block definition **double_block.ini**, or "default" to use the default single-block definition (used to reset block definition)]
- **nparc** *NPARC_info_file* [NPARC information file used to define flight conditions, etc., default is **NPARC_Info.npc**]
- **restart** *NPARC_restart_file* [NPARC restart file name, default is *case_name.rst*]
- **input** *NPARC_input_file* [NPARC input file name, default is *case_name.inp*]
- **output** *NPARC_output_file* [NPARC output file name, default is *case_name.out*]

Note: The "define" directive does not actually do anything other than define associations internal to ICEG2D.

3. **gen(erate)** – generates something
   - **dist(ribution)** – generates distribution using defined information
   - **grid** – generates grid using defined information
   - **input** – generates NPARC input file
   - **restart** – generates NPARC restart file
   - **solution** – runs NPARC as a batch job

4. **display** – graphically displays something
   - **dist(ribution)** – displays surface distribution (uses FAST)
   - **grid** – displays grid (uses FAST)
   - **weight** – displays smoothed weight function used in distribution (uses xmgr)

5. **show** – shows current file definition associations

6. **ls** – executes the Unix command "ls –la"
7. **help** – displays a help message

8. **exit, quit, bye** – exits ICG2D

Additional features of ICEG2D include:

- The user can start at any step of the process. For example, if a surface definition already exists, the user can start the process by specifying the file containing the surface distribution and then generate the grid.
- Results of each part of the process are saved automatically either using the default name based on the case name or by specification of the name directly.
- In the interactive mode, the user can display the distribution of points on the surface as well as the generated grid in a pop-up window with translate and zoom capability.
- The program generates a history file in ICEG2D.HST

The batch mode of ICEG2D is invoked using

```
SysPrompt% ICEG2D < case#1.txt > case#1.out
```

where `case#1.txt` is a text file containing the ICEG2D directives and `case#1.out` is the redirected output from ICEG 2D. The interactive mode is invoked using

```
SysPrompt% ICEG2D
```

and entering the ICEG2D directives from the ICEG2D prompt. Sample sessions for the batch and interactive modes are included in Section 7.

ICEG2D also prints informational messages to *stdout* indicating which associations are in effect, etc. Also printed are responses to the various ICEG2D directives.

3. Module ICE_DIST_2D

The quality of the surface distribution is critical to the ultimate success of a CFD simulation. This is particularly true for iced airfoils. Chung, et. al. [12] used an interactive software package, TURBO-GRD [13], to smooth and distribute points on a given iced airfoil surface. In [12], the number of control points could be decreased to reduce the surface complexity, i.e., smooth the surface.

The ICE_DIST_2D module is designed to generate a grid-ready surface definition for an iced airfoil with a high degree of automation. The surface definition may have been generated by an analysis program such as LEWICE [10] or obtained experimentally from the IRT [11]. The basic philosophy employed here is to represent an acceptable surface as faithfully as possible for a given number of points. Clustering points in regions of high curvature ensures geometric fidelity of the iced airfoil surface definition. At no time during the process is any explicit smoothing applied to the airfoil surface definition. However, as discussed below, the airfoil surface is represented internally as a NURBS curve. When the points are distributed on the surface, some smoothing is present due to the fact that the NURBS representation does not reproduce slope discontinuities in the surface description [14]. However, this smoothing is present once a NURBS representation is used to describe the geometry and is not a function of the distribution algorithm.

The module ICE_DIST_2D is written in C and utilizes the GGLib [2] geometry/grid library. ICEG2D passes four file names as command line arguments to ICE_DIST_2D: the iced airfoil input file, the clean airfoil input file, the output file for the surface distribution, and the distribution parameter file. These file names are defined using the associations made within ICEG2D. In the sections that follow, the input for the module is described along with the methodology used to automatically distribute points on the airfoil surface. The module ICE_DIST_2D is executed using the ICEG2D directive "generate distribution."

3.1 ICE_DIST_2D Input

The input for ICE_DIST_2D consists of a file containing the iced airfoil definition, a file containing the clean airfoil definition, and a file containing the parameters for the namelist DISTRIBUTION. The iced airfoil definition file, associated with *iced_geom_file* in the ICEG2D framework, is an ASCII file consisting of the integer number of points employed in the surface definition (*n_points_iced*) followed by *n_points_iced* (x,y) data pairs defining the surface. The data may or may not represent a full, closed airfoil definition. The algorithm to merge partial, iced airfoil data with full, clean airfoil data is described below. The clean airfoil file, associated with *clean_geom_file*, is defined similarly with the integer number of points employed in the surface definition (*n_points_clean*) followed by *n_points_clean* (x,y) data pairs defining the clean airfoil. The clean airfoil definition must be a full airfoil definition. **In both files, the surface is defined starting at the trailing edge on the lower surface proceeding clockwise to the trailing edge on the upper surface. The trailing edge point must be**

repeated as the first and last points in the file. If the trailing edge is not closed, the trailing edge point is placed at the average position of the upper and lower surface trailing edge location.

Input parameters used in the surface distribution are defined using a FORTRAN namelist. The namelist DISTRIBUTION is associated with *dist_parm_file*. The variables and their default values are listed in Table 3 and are described in detail in the appropriate section.

Namelist Variable	**dist_smooth.ini**	**dist_rough.ini**
prune	1.0	1.0
order	3	3
lower_surface_split_point	0.4	0.4
upper_surface_split_point	0.1	0.1
rpower	6.0	8.0
n_points_aft_lower_region	61	61
n_points_nose_region	151	251
n_points_aft_upper_region	101	101
int_max	2000	4000
n_smooth	200	800
cluster	2.0	2.0
max_min_ratio	5.0	5.0
delta_s_te	0.005	0.005

Table 3. Namelist DISTRIBUTION Variables

3.2 Merging and Pruning Algorithms

Data from the IRT is obtained in terms of partial iced airfoil shapes [15]. It is therefore necessary to devise an algorithm to merge partial iced airfoil data with a clean airfoil definition to obtain a complete airfoil. Additionally, the iced airfoil data often contains twists or loops in which the curve describing the surface actually loops back onto itself. Here, the operation removing these undesirable loops is termed pruning. Both algorithms are described below.

In developing the merging algorithm, there are two primary issues that must be addressed:
1) The locations on the clean airfoil definition where the iced airfoil should be merged must be determined and 2) the curves used to form the transitions between the partial iced surface and the clean surface must be specified. It should be noted that the approaches used to address these issues are based purely on heuristics. The primary objective is to ensure that the merging process introduces no significant discontinuities.

The merging algorithm implemented here is based on identifying the points in the clean airfoil surface definition closest to the two endpoints of the partial iced airfoil definition. Once these two points are located, one for each endpoint, the slope of the iced airfoil surface at each endpoint is used to estimate the intersection of the ice surface with the

clean airfoil surface. The clean airfoil surface is then deleted between the intersection points. C^1 continuous Hermite interpolation [16] is then used to generate transition curves between the two surfaces.

Figure 1 shows examples of the transitions for a merged surface definition corresponding to IRT data for the GLC305 airfoil section denoted as case 904EXP [17]. This case was artificially created by deleting points from a full surface definition. As can be seen from the figure, the transition on the lower surface is smooth. A close up of the region near the intersection between the horn and the upper surface is shown in Figure 2. Again, the transition is smooth and maintains the surface continuity.

The pruning algorithm is also based on heuristics. The basis for the pruning algorithm is the observation that when "looping" occurs the following inequality is satisfied:

$$\left\|\mathbf{r}_{i+k} - \mathbf{r}_i\right\| \le \beta \left\|\mathbf{r}_{i+1} - \mathbf{r}_i\right\| \qquad for \qquad k > 1$$

where β is a proportionality constant of $O(1)$ defined via the DISTRIBUTION namelist parameter *prune*. The above equation states that looping is a concern if a nonadjacent point, $\mathbf{r}_{i+k}$, is closer to the point under consideration, $\mathbf{r}_i$, than a specified tolerance that is based on the distance to the adjacent point, $\mathbf{r}_{i+1}$. The algorithm implemented here simply eliminates the points contained within the loop $\mathbf{r}_i$ through $\mathbf{r}_{i+k}$.

Figure 3 shows a pruned surface generated using $\beta = 1$. The surface definition corresponds to the IRT data denoted as case 904EXP [17]. The pruning algorithm described here generates a usable surface definition from a clearly unacceptable surface. At this time, the effects of the pruning have not yet been quantified. However, it seems unlikely that any negative effects would be significant.

3.3 Internal Surface Description

Once a full surface description is obtained, an internal representation of the surface must be generated. Routines from the geometry/grid library GGLib [2] are used to define a NURBS representation for the iced airfoil surface. The parametric description of the NURBS curve is given by

$$\mathbf{P}(u) = (x(u), y(u))$$

where u is the parametric variable. The namelist variable *order* defines the order of the NURBS approximation. An inversion routine from GGLib is used to compute the control points needed to produce the specified surface. It should be noted that this inversion is the single most time-consuming element of the process aside from the CFD solution itself but requires less than one minute on typical workstations. Using another GGLib routine, the

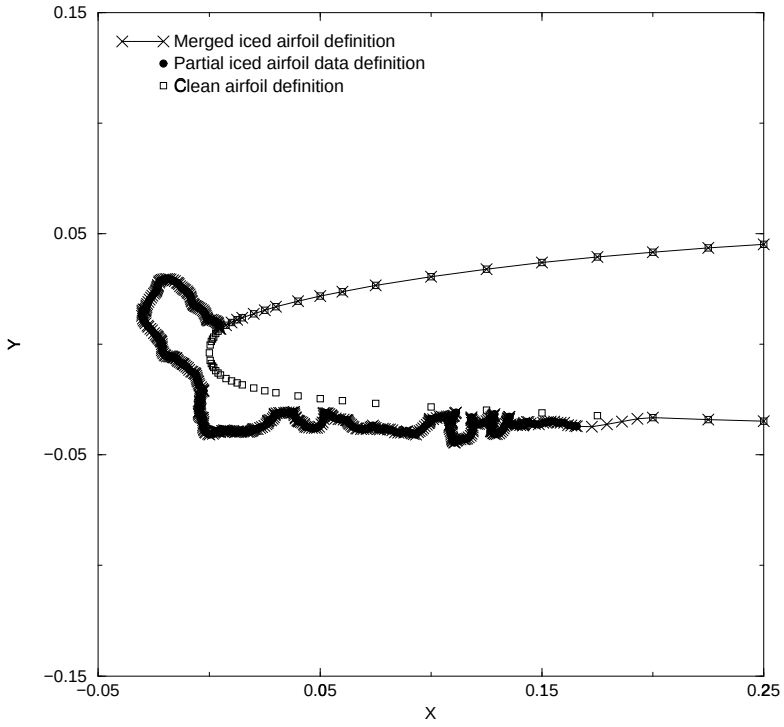


Figure 1. Demonstration of Merging Algorithm

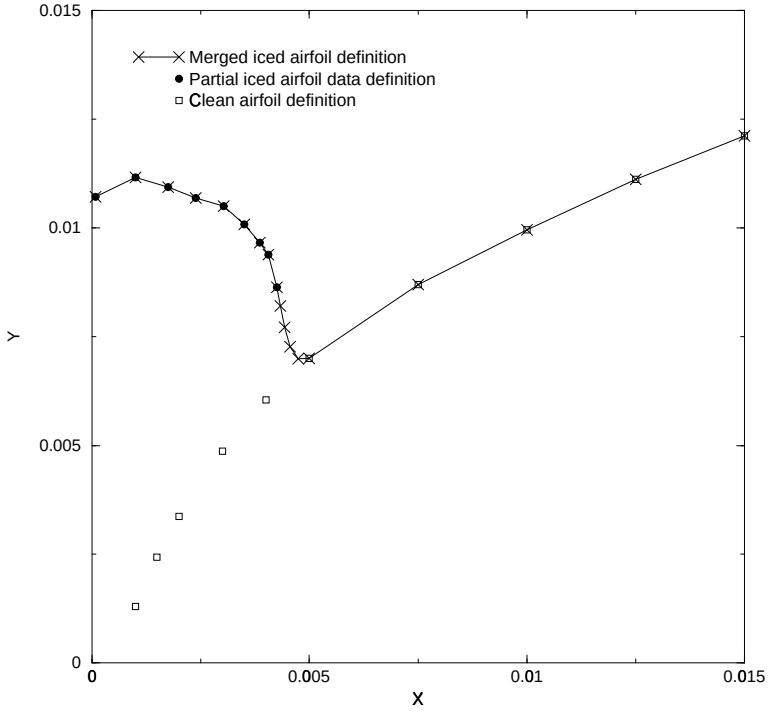


Figure 2. Merging in Region Near Horn/Upper Surface Intersection

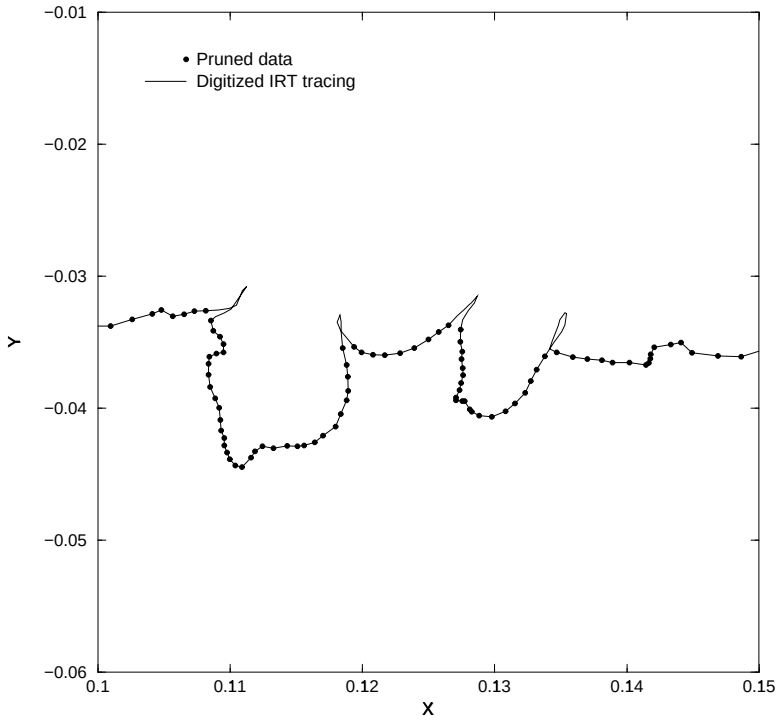


Figure 3. Demonstration of Pruning Algorithm

NURBS description is then split into three parts – the aft lower surface $\mathbf{P}_L(u)$, the nose region $\mathbf{P}_N(u)$, and the aft upper surface $\mathbf{P}_U(u)$ - based on the DISTRIBUTION namelist variables *lower_surface_split_point* and *upper_surface_split_point* which represent x/c locations on the lower and upper surfaces respectively as shown in Figure 4. These split points should be defined such that the ice accretion is contained within the nose region. Since runback usually occurs only on the lower surface when the airfoil is at positive angle of attack, the upper surface split point is typically set significantly closer to the leading edge than the lower surface split point. The methods used to distribute points in each segment are described below.

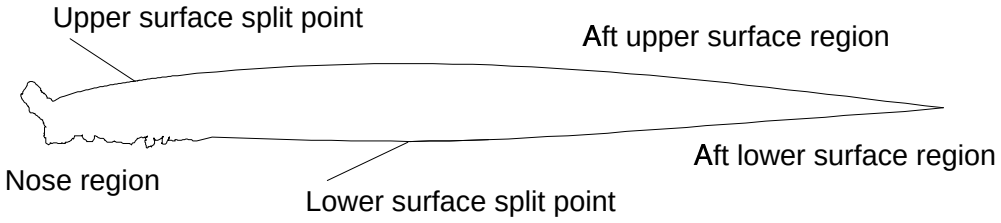


Figure 4. Nomenclature for NURBS Description of Iced Airfoil

3.4 Surface Point Distribution

The first step in distributing points on the airfoil surface is to determine the number of points to be used on each segment. The philosophy used here is to make the total number of points be a function of the length of the curve that describes the iced airfoil. In this way, a larger ice accretion is defined using more points. Currently, the only mechanism to account for a "jagged" surface in which the length of the curve does not change significantly is to use the "define surface rough" directive. This directive associates the file **dist_rough.ini** with *dist_parm_file*. The default values for the variables in the DISTRIBUTION namelist for smooth and rough surfaces are shown in Table 3.

The number of points on the iced airfoil surface is based on a function of the ratio of the iced airfoil surface length (s_{iced}) to the clean airfoil surface length (s_{clean})

$$\left(\frac{n_{iced}}{n_{clean}}\right) = \left(\frac{s_{iced}}{s_{clean}}\right)^{rpower}$$

where

$$n_{clean} = \text{n\_points\_aft\_lower\_region} + \text{n\_points\_aft\_upper\_region} + \text{n\_points\_nose\_region}$$

and *rpower* is specified via the DISTRIBUTION namelist. To accommodate the method used to generate double-block grids, n_{iced} is automatically monitored to ensure that it is an odd number. The increase in points from n_{clean} to n_{iced} is assumed to occur in the iced region. The number of points in the aft lower and aft upper segments is defined via namelist input. It is assumed that the nose region is defined such that it is the only region in which ice accretion occurs. Therefore,

$$N = n_{iced} - \text{n\_points\_aft\_lower\_region} - \text{n\_points\_aft\_upper\_region}$$

where N is the resulting number of points in the nose region.

The distribution of points in the nose region is based on an equidistribution principle [16]. The approach used here is to distribute points on the NURBS description of the nose of the airfoil, $\mathbf{P}_N(u)$, using a weight function based on surface curvature. For a distribution of *N* points along a curve of length s_{iced}, the location of the i^{th} point is computed using equidistribution from

$$\int_0^{s_i} w(s)\,ds = \frac{i-1}{N-1}\int_0^{s_{iced}} w(s)\,ds$$

where w(s) is the weight function and s is the arc length along the curve [16]. The objective is to determine the value of the integrand s_i that satisfies the equality for each value of *i*. A direct result of using the equidistribution formulation is that in regions

where the positive weight function is large, the mesh spacing is correspondingly small. Therefore, if the weight function is based on surface curvature, points will be clustered in regions of higher curvature leading to a better representation of the surface. For the problem at hand, the weight function is known explicitly in terms of the parametric arc length u. Therefore, using a change of variable, the equidistribution relation becomes

$$\int_0^{u_i} w(u)\left(\frac{ds}{du}\right)du = \frac{i-1}{N-1}\int_0^1 w(u)\left(\frac{ds}{du}\right)du$$

and (ds/du) is defined using

$$\frac{\mathrm{ds}}{\mathrm{du}} = \sqrt{\mathbf{P}'_{\mathrm{N}}(\mathrm{u})\cdot\mathbf{P}'_{\mathrm{N}}(\mathrm{u})}$$

where $\mathbf{P}'_N(u)$ is computed using appropriate GGLib routines. The integral is evaluated by distributing *int_max* (specified via the DISTRIBUTION namelist) equally spaced integration nodes on the NURBS representation $\mathbf{P}_{\mathrm{N}}(\mathrm{u})$ in parametric space and performing a piecewise linear fit of the integrand. Using the integral formulation eliminates many of the problems associated with the finite-difference version of the equidistribution equation. In particular, no iteration is required to obtain the point distribution. Routines from GGLib are then used to evaluate the resulting (x,y) locations on $\mathbf{P}_{\mathrm{N}}(\mathrm{u})$.

Using appropriate GGLib routines, the surface curvature is computed using

$$K(u) = \frac{\left|\ \left(\mathbf{P}'_N(u)\cdot\mathbf{P}'_N(u)\right)\mathbf{P}''_N(u) - \left(\mathbf{P}''_N(u)\cdot\mathbf{P}'_N(u)\right)\mathbf{P}'_N(u)\ \right|}{\left(\mathbf{P}'_N(u)\cdot\mathbf{P}'_N(u)\right)^2}$$

A clustering function is now defined at each of the integration nodes

$$c(u_j) = K(u_j)^{(1-1/\alpha)}$$

where $K(u_j)$ is the curvature and $\alpha \geq 1$ is a clustering parameter defined by the DISTRIBUTION namelist variable *cluster*. The general effect of decreasing α is to increase the strength of the peaks of the normalized curvature, i.e., to increase the variation in curvature and increase the clustering in the point distribution.

The second step in formulating the weight function is to smooth the clustering function through repeated application of

$$\bar{c}(u_j) = \frac{1}{4}\left(c(u_{j-1}) + 2c(u_j) + c(u_{j+1})\right), \quad j = 2, \mathrm{int\_max} - 1$$

It is important to note that the geometry is not being smoothed - only the clustering function used to define the weight function in the equidistribution scheme is being

smoothed. The purpose of the smoothing is to reduce the variation of the surface point distribution in regions of rapid variation in curvature. Finally, the normalized weight function is defined in terms of the smoothed clustering function using

$$w(u) = \frac{\max(\bar{c}(u), \hat{c}_{min})}{\bar{c}_{max}}$$

where $\hat{c}_{min} = \max(\bar{c}_{max} / \sigma, \bar{c}_{min})$. $\hat{c}_{min}$ serves to keep the ratio of the maximum spacing to the minimum spacing in the nose region less than the tolerance σ which is specified via the DISTRIBUTION namelist variable *max_min_ratio.*

Points are distributed on the aft upper and aft lower surface segments, $\mathbf{P}_U(u)$ and $\mathbf{P}_L(u)$ respectively, using a hyperbolic tangent stretching function contained in GGLib. The spacing at the inboard end of each segment is set to match the spacing obtained from the nose region to ensure a smooth point distribution in the transition region. The spacing at the trailing edge is specified using the variable *delta_s_te* from the DISTRIBUTION namelist.

3.5 Demonstration of Distribution Algorithm

It is now appropriate to present results for the point distribution algorithm as well as demonstrate the effects of various parameters that influence the surface distribution. Issues related to the fidelity of the geometric representation of the iced airfoil surface can easily be addressed by critical examination of several representative configurations. Ultimately, the issue of surface grid quality can be addressed only by performing numerical calculations and comparing the results with experimental data. These issues are addressed in Section 6.

The first case considered here is a relatively smooth shape generated by LEWICE for the GLC305 airfoil section and is designated case 204LEW [17]. Figure 5 shows a plot of the normalized weight function versus the parametric coordinate u in the iced region and demonstrates the effects of smoothing on the weight function. The parametric coordinate on this segment of the surface varies from a value of zero at the lower surface split point to a value of unity at the upper surface split point. Only a portion of the iced region of the airfoil is shown here. As can be seen from the figure, the general effect of smoothing is to reduce the rapid variation in the weight function. This ultimately results in a smoother point distribution on the surface as points are clustered in regions of high curvature rather than to discrete points of high curvature. This result suggests applying additional smoothing to rough or jagged ice shapes. This is the approach used in ICE_DIST_2D. A second distribution parameter file, **dist_rough.ini**, is used when *surface* is defined using "def surface rough." Table 3 contains the default values for DISTRIBUTION namelist variables for both smooth and rough surfaces. Also shown in the figure is the effect of the

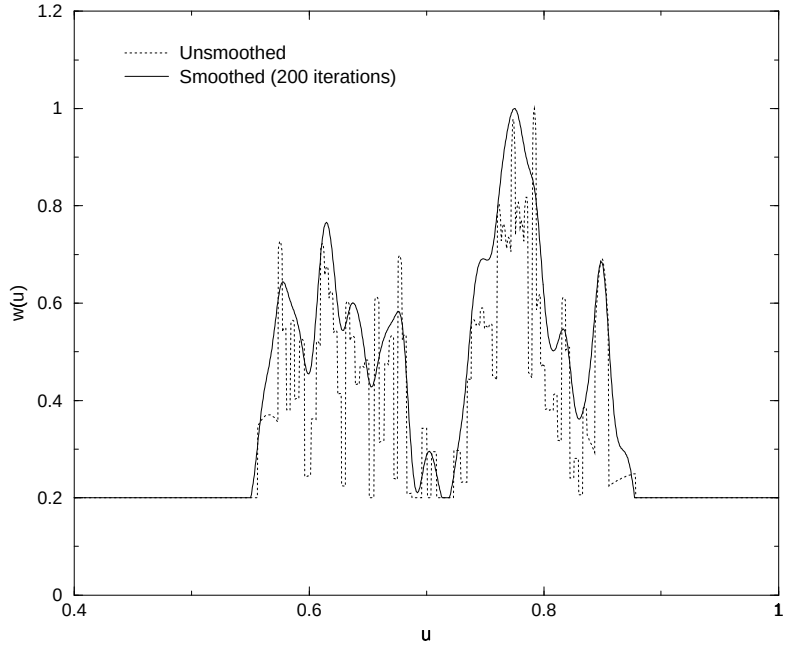


Figure 5. Effect of Smoothing on Normalized Weight Function

variable *max_min_ratio*. In this case, the ratio is set to 5.0 so that an artificial "floor" of 0.2 is placed on the normalized weight function.

Figure 6 shows the effect of the parameter *cluster* on the normalized weight function. The general effect of decreasing the parameter is the enhancement of peak values relative to the maximum peak. Given a fixed number of points, the net effect on the surface point distribution is the enhancement of clustering at the other peaks shown in the figure at the expense of clustering at the primary peak and other regions. In the extreme case, *cluster=1*, the weight function is unity everywhere and equal spacing results.

Figure 7 shows a plot of the surface distribution and the original iced airfoil data for case 204LEW [17]. As can be seen from the figure, the overall geometric fidelity of the surface is well maintained and points are clustered in regions of higher surface curvature. However, one point should be made regarding surface slope discontinuities. Note that in the region where the ice horn intersects the upper surface of the airfoil the surface distribution is at least C^1 continuous (actually it is C^2 continuous) while the original data suggests only C^0 continuity. This condition will occur wherever the initial surface is only C^0 continuous. Note, however, that the curve does pass through the corner point. In effect, the surface is implicitly smoothed while maintaining the interpolating nature of the NURBS approximation. It should be noted that the straight-line segments in the original surface distribution near the right edge of the figure occur because the spacing between the points increases dramatically. The NURBS approximation in ICE_DIST_2D interpolates these points using a smooth curve rather than a line segment.

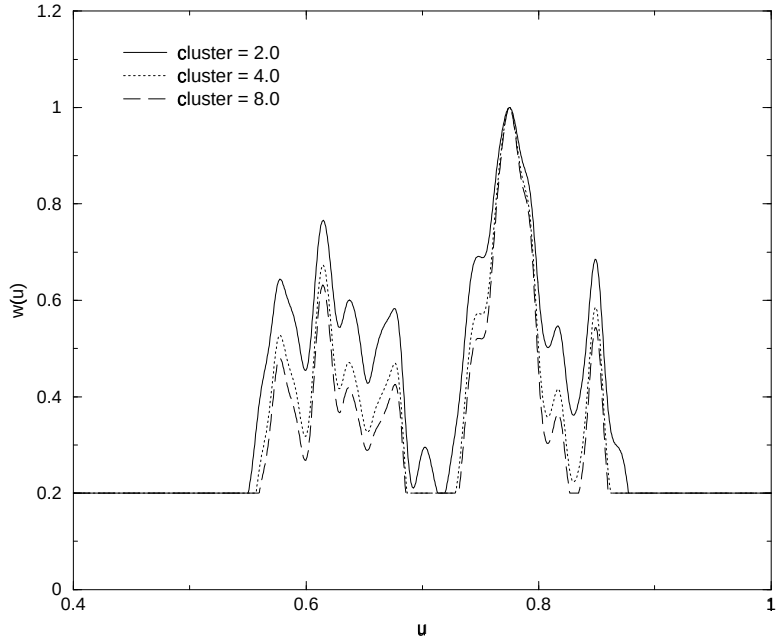


Figure 6. Effect of Clustering Parameter on Normalized Weight Function

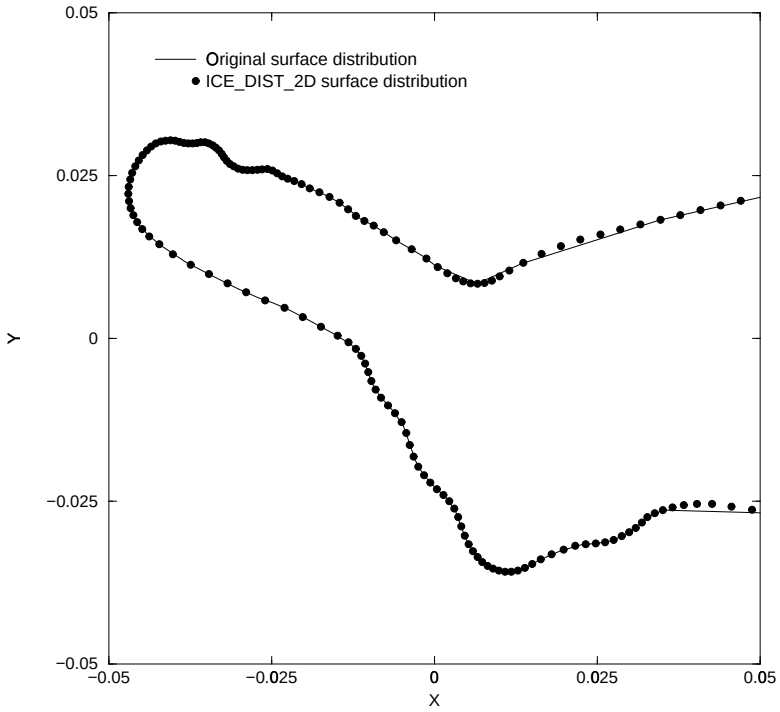


Figure 7. 204LEW Surface Definition

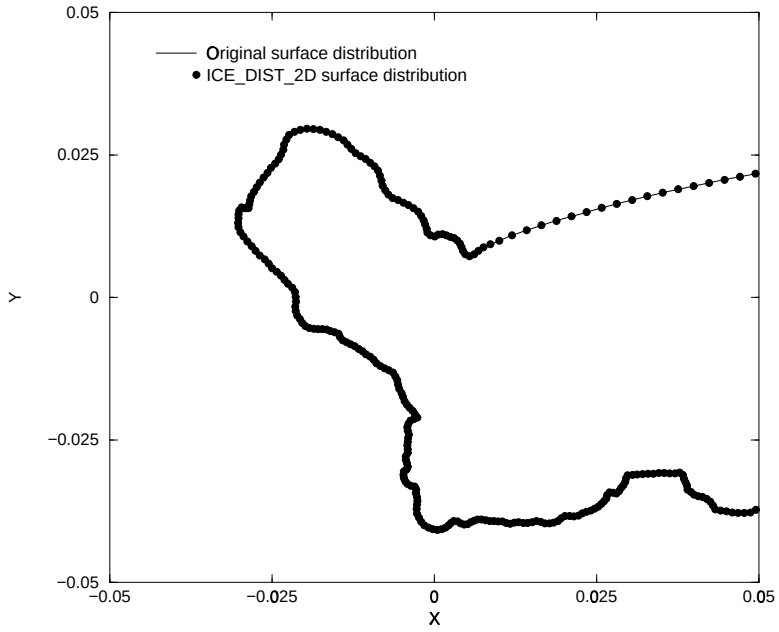


Figure 8. 904EXP Surface Definition

Figure 8 shows a plot of the surface distribution and the original iced airfoil data for the case 904EXP obtained from IRT data [17]. This case is considerably more challenging. The surface distribution again shows point clustering in regions of higher curvature. However, the clustering is not as apparent as it is in Figure 7. This occurs because of the overall roughness of the ice shape. Essentially, there are fewer distinct features about which to cluster. Additionally, 904EXP has a significant, jagged accretion of ice on the lower surface. The lower surface accretion requires additional points to represent this portion of the surface and effectively removes points from the horn region.

It should be emphasized that the surface distribution is generated automatically. For most cases, no user intervention is required.

3.6 ICE_DIST_2D Output

The output from the ICE_DIST_2D module consists of a file containing the surface data and informational messages to *stdout*. The three airfoil segments are concatenated and output to a file in the PLOT3D, multizone, formatted, two-dimensional, grid format [9]. The surface distribution may be viewed using the "display distribution" directive from ICEG2D. The output file name is associated with *dist_file* with the default being *case_name.dst*. A sample of the information printed to *stdout* where *case_name* has been set to "623exp" and *surface_type* has been set to "rough" is shown below:

```
Generating distribution using:

Iced geometry file        - 623exp.dat
Clean geometry file       - 623exp.cln
Parameter definition file – dist_rough.ini
```

```
Distribution output file  - 623exp.dst


Begin surface distribution calculation.

prune                     = 1.000000
order                     = 3
n_points_aft_lower_region = 61
n_points_nose_region      = 251
n_points_aft_upper_region = 101
lower_surface_split_point = 0.400000
upper_surface_split_point = 0.100000
delta_s_te                = 0.005000
cluster                   = 2.000000
rpower                    = 8.000000
max_min_ratio             = 5.000000
int_max                   = 4000
n_smooth                  = 800

*** WARNING *** Open trailing edge found for clean airfoil.
TE location set at average of upper and lower points.

*** WARNING *** Open trailing edge found for iced airfoil.
TE location set at average of upper and lower points.

Ratio = 1.052840
Number of points on airfoil = 621

ds(max)/ds(min) = 5.373643

Lower surface
   Trailing edge - ds(gen) = 0.005098, ds(spec) = 0.005000
   Split point   - ds(gen) = 0.003072, ds(spec) = 0.003003

Upper surface
   Split point   - ds(gen) = 0.003039, ds(spec) = 0.003003
   Trailing edge - ds(gen) = 0.005049, ds(spec) = 0.005000

Surface distribution generation complete!
```

The output consists of an inventory of relevant file associations followed by an echo of the namelist DISTRIBUTION. The user is then notified that an open trailing edge was detected and action was taken. The ratio s_{iced}/s_{clean} is then output along with the number of points on the airfoil surface n_{iced}. Diagnostic information follows showing an approximate maximum to minimum spacing ratio based on cord length (as opposed to arc length) in the nose region and a comparison of the specified and generated spacings at the end points of the aft upper segment and aft lower segment. Finally, a message is printed indicating successful completion of the surface definition.

4. Module ICE_GRID_2D

Automated grid generation for complex configurations is an, as yet, unsolved challenge. Even grids for the topologically simple single-element iced airfoils are challenging because of concave regions occurring on the surface of the airfoil. These regions pose difficulties for algebraic or elliptic grid generation algorithms because of the difficulty in placing corresponding points on the outer boundary of the grid. One approach that obviates this problem is to use a marching technique to generate the grid. In this context, a marching scheme uses the surface distribution as the initial data curve and generates a grid layer by layer until a specified number of layers have been generated. Typically, the marching distance distribution, i.e., the thickness of each layer, is specified. Both parabolic [3-7] and hyperbolic [18] schemes have been used for marching grid generation algorithms. In this effort, a parabolic scheme was chosen because of the ability to generate solution adaptive grids [4,5]. It should be noted that solution adaptive grid capability is scheduled for implementation in Years 2 and 3.

The module ICE_GRID_2D is written in FORTRAN90 with a C driver routine and utilizes dynamic memory allocation. ICEG2D passes the case name and four file names as command line arguments to ICE_GRID_2D: the airfoil surface definition file, the output file for the grid, the grid generation parameter file, and the block definition parameter file. These file names can be defined using the default associations in ICEG2D or by user specification. In the sections that follow, the input for the module is described along with the methodology used to automatically generate a single- or double-block, C-type grid. The module ICE_GRID_2D is executed using the directive “generate grid.”

4.1 ICE_GRID_2D Input

The input to ICE_GRID_2D consists of a file containing the data for the namelist GRID_PARAMETERS and a file containing data for the namelists NUMBER_OF_BLOCKS and BLOCK_PARAMETERS. The airfoil surface definition file, associated with *dist_file*, is required to define the initial surface for the grid generation algorithm. The case name is used to generate an identifying banner in the output from the module to *stdout*. The variables in the various namelists and their default values are listed in Tables 4-7 below and are defined in the appropriate sections.

Namelist Variable	**Default Value**
itMax	20
nOrder	2
omega	1.0
addedDissipation	1.0
iWake	40
distWake	15.0

Table 4. Namelist GRID_PARAMETERS Variables

Namelist Variable	single_block.ini	double_block.ini
nBlocks	1	2

Table 5. Namelist NUMBER_OF_BLOCKS Variables

Namelist Variable	single_block.ini
distFirstPointLE_nl	$1.0x10^{-6}$
distFirstPointTE_nl	$1.0x10^{-6}$
distFirstPointDSB_nl	$1.0x10^{-5}$
distOuterBoundary_nl	15.0
xTransLower_nl	0.5
xTransUpper_nl	0.5
jMax_nl	101

Table 6. Namelist BLOCK_PARAMETERS Variables – Single-Block Grid

Namelist Variable	Block #1	Block #2
distFirstPointLE_nl	$1.0x10^{-6}$	Computed
distFirstPointTE_nl	$1.0x10^{-6}$	Computed
distFirstPointDSB_nl	$1.0x10^{-5}$	Computed
distOuterBoundary_nl	1.0	14.0
xTransLower_nl	0.5	NA
xTransUpper_nl	0.5	NA
jMax_nl	71	21

Table 7. Namelist BLOCK_PARAMETERS Variables – Double-Block Grid

4.2 Blocking Strategies

The ICEG2D framework allows for two different structured grid configurations: a single-block topology and a double-block topology. The single-block grid is generated directly using the techniques outlined below. The inner block of the double-block grid, designated as block number one, is treated in the same manner. The outer block grid generation requires the specification of an initial surface. The outer block is assumed to overlap the last two cells of the inner block. The outer-block initial surface is defined using a surface generated by taking the coordinates of the first and last points and every other point in between on the surface *j=jMax-2* on the inner grid. Recall that the number of points on a *j=constant* surface in the inner grid is forced to be odd, so this approach will always work.

In general, the file *block_parm_file* will consist of (*nBlocks+1*) different namelists – a single occurrence of the namelist NUMBER_OF_BLOCKS plus the *nBlocks* occurrences of the BLOCK_PARAMETERS namelists - one for each block in topology.

4.3 Wake Cut Definition

Because of improved resolution at the airfoil trailing edge, a C-type grid is used in all cases. Therefore, it is necessary to define a wake cut that extends from the trailing edge of the airfoil to the downstream boundary of the computational domain. Here, it is assumed that the cut is parallel to the *x*-axis. The distance to the downstream boundary is specified via the namelist GRID_PARAMETERS using the variable *distWake*. The number of points defining the cut is given by *iWake*. Note that the trailing edge point is considered a wake point so that the number of points downstream of the trailing edge is *iWake-1*. Therefore, the total number of points on the *j=1* surface is given by

$$iMax = n_{iced} + 2 \times (iWake - 1)$$

The distribution of points on the cut is defined using a hyperbolic tangent stretching function [19]. The first point downstream of the trailing edge is located a distance equal to the distance from the trailing edge to the first point upstream of the trailing edge on the airfoil surface. This distance, the distance to the downstream boundary, and the number of points on the cut are sufficient to compute the distribution using the hyperbolic tangent function.

4.4 Normal Distance Distribution

The normal distance distribution is generated using the hyperbolic tangent stretching function defined in [19]. The distance to the first point off the wall is specified along with the distance to the last point and the total number of points. The hyperbolic tangent function was selected because it was shown to induce the least overall truncation error of several stretching functions considered.

As implemented here, the distance to the first point off the wall is specified at three locations: the leading edge of the airfoil *distFirstPointLE_nl*, the trailing edge of the airfoil *distFirstPointTE_nl*, and the downstream boundary of the computational domain *distFirstPointDSB_nl*. Using these three values and the transition points *xTransLower_nl* and *xTransUpper_nl*, linear interpolation is used to define the distance to the first point for each point on the initial data surface. For the upper and lower surfaces

$$\delta_i^1 = \begin{cases} \delta^{LE} & , \quad x_{LE} \le x_i \le x_{TRANS} \\ \delta^{LE} + \left(\dfrac{x_i - x_{TRANS\_U,L}}{x_{TE} - x_{TRANS\_U,L}} \right) \left(\delta^{TE} - \delta^{LE} \right) & , \quad x_{TRANS} < x_i \le x_{TE} \\ \delta^{DSB} + \left(\dfrac{x_i - x_{TE}}{x_{DSB} - x_{TE}} \right) \left(\delta^{DSB} - \delta^{TE} \right) & , \quad x_{TE} < x_i \le x_{DSB} \end{cases}$$

where δ_i^1 is the distance to the first point off the wall at x_i. These values are then smoothed to eliminate the propagation of slope discontinuities from the boundary into the interior of the grid. This approach is used to give flexibility regarding the normal mesh distribution at the leading edge, the trailing edge, and the downstream boundary. Then, the distance to the outer boundary *distOuterBoundary_nl* and number of points *jMax_nl* are used to compute the normal distance distribution $\delta_i^j, j = 1, jMax$ at each point on the initial data surface.

In the case of the double-block configuration, the interior block is treated in an identical manner as above. The distance to the first point in the outer block is based on the spacing in the overlap region in the inner block to ensure a good match with the inner block grid.

4.5 Parabolic Grid Generation

In the parabolic method [3-7], a reference grid is utilized to make the marching problem well posed. Starting from the initial data surface, two reference grid surfaces are generated. The Poisson grid generation equations are applied to points on the first surface of the reference grid. This step is equivalent to smoothing the reference grid. The second surface is updated after each iteration of the Poisson grid equations. Note that the second surface of the reference grid is discarded once the desired number of smoothing iterations is complete. It should also be noted that the resulting smoothed grid still exhibits many of the characteristics of the reference grid.

As is customary [16], a transformation of the form

$$\xi = \xi(x, y)$$
$$\eta = \eta(x, y)$$

is defined and the inverse transformation

$$x = x(\xi, \eta)$$
$$y = y(\xi, \eta)$$

is assumed to exist. The initial data surface is assumed to be defined by a η =*constant* surface and η is taken to be the marching direction.

4.5.1 Reference Grid Definition

The reference grid is generated algebraically to be locally orthogonal. The first surface of the reference grid in layer *j* is defined using

$$\boldsymbol{r}_{i,1}^{j} = \boldsymbol{r}_{i,0}^{j} + \delta_i^j \boldsymbol{n}_{i,0}^{j}$$

where $\boldsymbol{r}_{i,0}$ is the position vector to the point located at *(i)* on the initial data surface (0 subscript), $\boldsymbol{r}_{i,1}$ is the position vector to the point *(i)* on the first surface of the reference grid, δ_i is the specified distance distribution, and $\boldsymbol{n}_{i,0}$ is the unit surface normal. The second surface of the reference grid in layer *j* is generated using

$$\boldsymbol{r}_{i,2}^{j} = \boldsymbol{r}_{i,1}^{j} + \delta_i^{j+1} \boldsymbol{n}_{i,1}^{j}$$

Although not discussed here, it is possible to adjust the reference grid as the grid marches away from the initial data surface to meet specified outer boundary points.

4.5.2 Grid Smoothing

The equations used to smooth the reference grid are the standard Poisson equations [16] used in grid generation with the assumption of grid orthogonality, i.e., g_{12} is zero:

$$g_{22}((1+\nu)\mathbf{r}_{\xi\xi} + \phi\mathbf{r}_{\xi}) + g_{11}(\mathbf{r}_{\eta\eta} + \psi\mathbf{r}_{\eta}) = 0$$

where

$$g_{11} = x_{\xi}^2 + y_{\xi}^2$$
$$g_{22} = x_{\eta}^2 + y_{\eta}^2$$

The partial derivatives in the Poisson equation are approximated using second-order central differences. The resulting system of equations is solved using line relaxation.

The form of the control functions plays an important role in determining the quality of the mesh. The form of the control functions used here does not include curvature effects [20] and is given by

$$\phi = -\frac{1}{g_{11}}\frac{\partial g_{11}}{\partial \xi}$$
$$\psi = -\frac{1}{g_{22}}\frac{\partial g_{22}}{\partial \eta}$$

The control function ϕ is computed using the initial data line and held constant along each $\xi = \text{constant}$ line. ψ can be approximated using the normal distance distribution by noting that $g_{22} = \delta^2$.

On occasion, it has been found that adding dissipation to the Poisson smoothing equation is necessary, particularly when generating meshes in strongly nonconvex regions [6,7]. This approach is similar in spirit to the approach taken by Chan and Steger [18] and is shown in detail for the ξ direction.

In regions where $g_{11} >> g_{22}$, the dominant term in the Poisson equation is the term containing the η derivatives. In nonconvex regions, the resulting lack of smoothing along η=constant lines can lead to mesh line crossing. The approach taken here is to include the additional smoothing term $(1+\nu)$ in the Poisson smoothing equation where

$$\nu = \sqrt{\frac{\max(g_{11}, g_{22})}{g_{22}}} \times f\left(\theta_\xi\right)$$

where

$$f\left(\theta_\xi\right) = \begin{Bmatrix} 1 & 0 \le \theta_\xi < \frac{\pi}{2} \\ \sin\theta_\xi & \frac{\pi}{2} \le \theta_\xi < \pi \\ 0 & \pi \le \theta_\xi \end{Bmatrix}$$

and θ_ξ is the angle between $\boldsymbol{r}_{i+1}$-$\boldsymbol{r}_i$ and $\boldsymbol{r}_{i-1}$-$\boldsymbol{r}_i$. The amount of dissipation may be increased or decreased via the GRID_PARAMETERS namelist variable *addedDissipation* that has a default value of unity.

Recall the objective here is not to solve the elliptic grid generation equations but rather use them as an intelligent smoothing mechanism. In practice, it has been found that reducing the grid point movement induced by the smoothing by two orders of magnitude results in a grid that is sufficiently smooth. This value corresponding to the number of orders of magnitude reduction in the residual is specified via the GRID_PARAMETERS namelist variable *nOrder*. Typically, this process requires the equivalent of fewer than ten iterations of an elliptic solver. Domains containing strongly non-convex regions require additional iteration. The maximum number of iterations is specified using *itMax* in the GRID_PARAMETERS namelist. It is also possible to adjust the relaxation factor `omega`. However, it has been determined that the best results are typically obtained for the default value of unity.

4.6 Demonstration of ICE_GRID_2D

ICEG2D was used to generate several representative grids. In all cases, the grids were generated using the surface definitions shown in the preceding section. These grids and surface distributions were generated automatically.

Figure 9 shows the full computational domain for a single-block grid generated around the 204EXP ice shape [17]. The grid dimensions are 517x101. Figure 10 shows the region of the grid near the airfoil leading edge. As can be observed in the figures, the grid distribution is relatively smooth on the surface and in the interior of the domain. Further, the grids show the coalescence of lines emanating from concave regions of the airfoil surface that is characteristic of all marching methods. Figures 11 and 12 show a double-block grid generated using the same surface distribution for the 204EXP ice shape and demonstrate the capability of ICEG2D to generate multi-block grids automatically. The dimensions of the inner and outer blocks are 517x71 and 259x21 respectively.

Figure 13 shows a region of a single-block grid generated for the 904EXP ice shape [17]. The right edge of this figure corresponds to a distance of approximately 15 percent of the chord aft of the leading edge of the airfoil. Although this case in considerably more challenging due to the ice accretion on the lower surface, the grid appears to be of high quality. The coalescence of lines is much more evident than in Figures 9-12 because of the severity of the concave regions. Figure 14 shows the region near the ice growths near the right side of Figure 13. Again a smooth variation in the grid is evident in the figure. The grid dimensions in this case are 829x101. It should be noted that there is significant skewness that occurs in less than 0.1 percent of the cells. This skewness is due to the strongly concave regions on the airfoil surface.

Again, it should be emphasized that both the single- and double-block grids are generated automatically. For most cases, no user intervention is required.

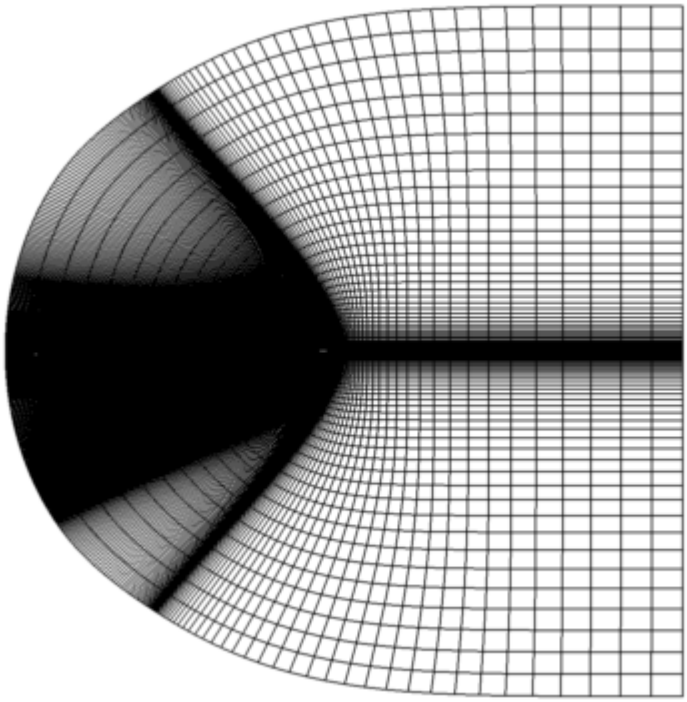


Figure 9. 204LEW Airfoil Grid

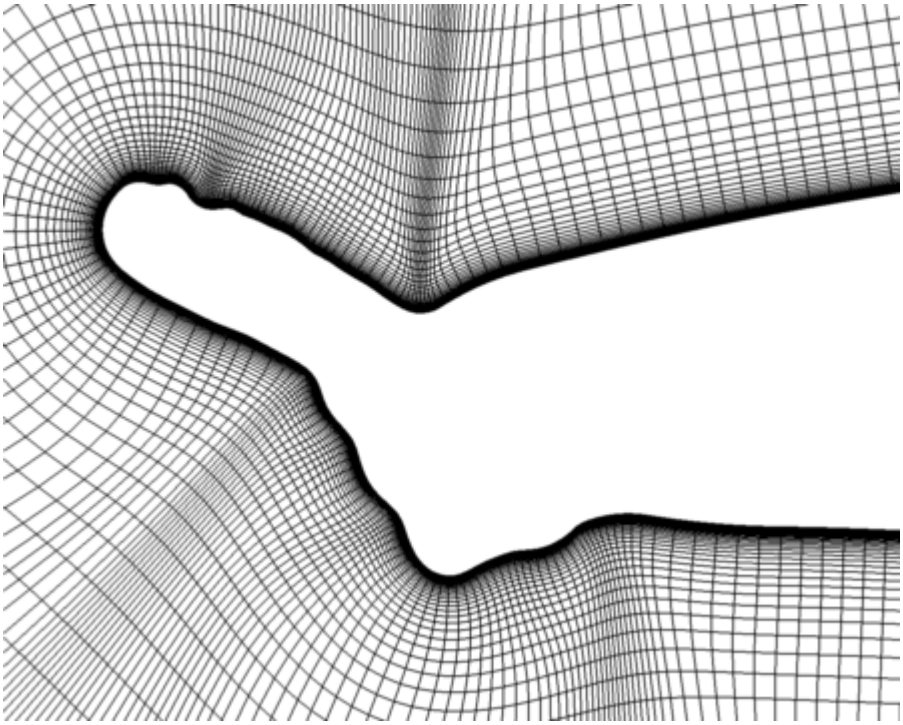


Figure 10. Nose Region of 204LEW Airfoil Grid

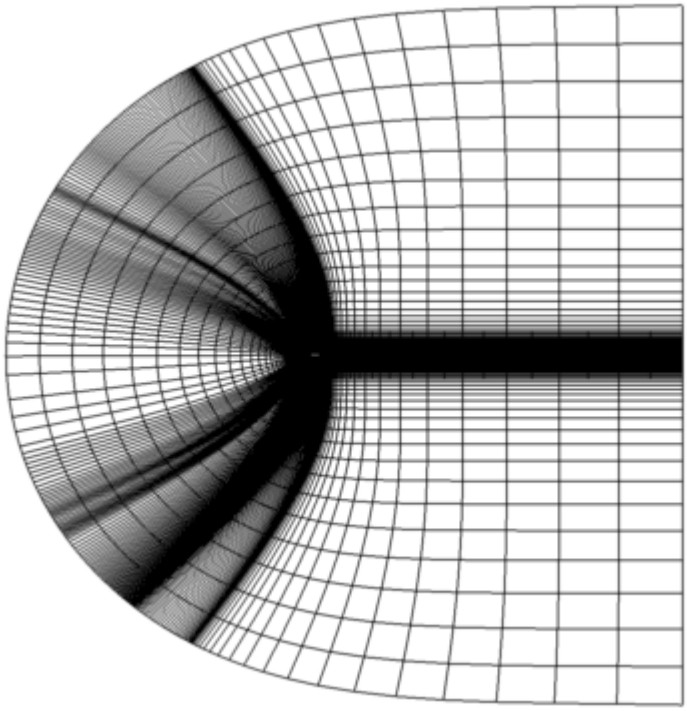


Figure 11. 204LEW Airfoil Grid – Double Block

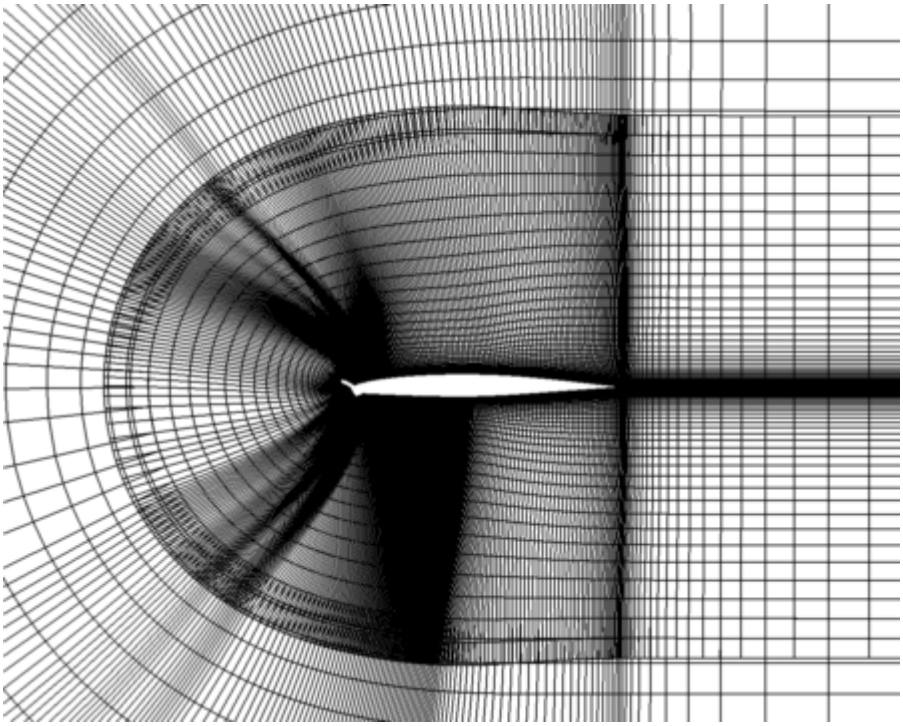


Figure 12. 204LEW Airfoil Grid – Double Block

Figure 13. 904EXP Airfoil Grid

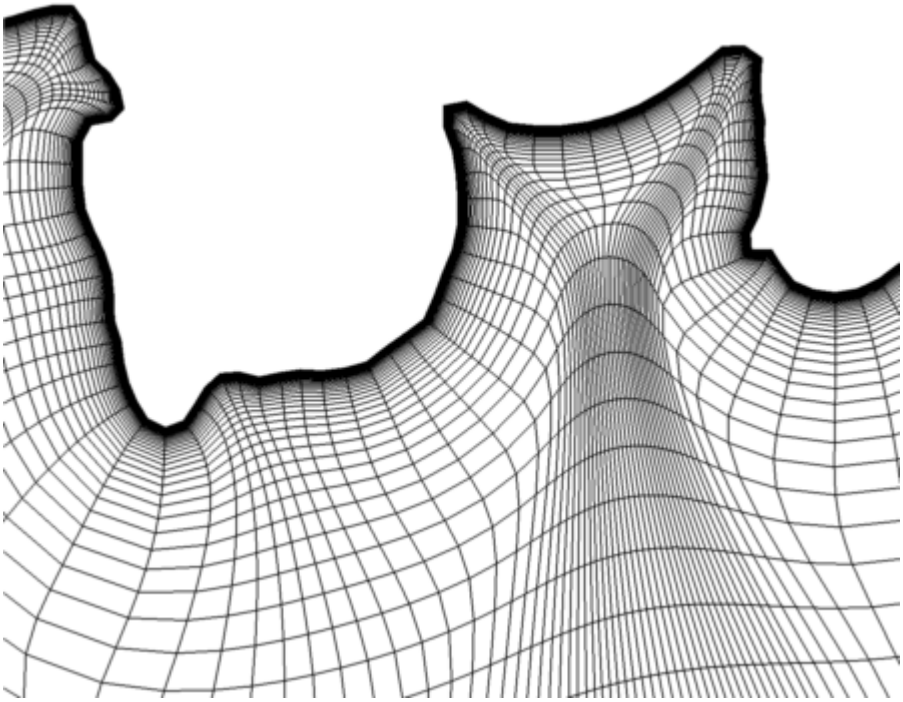


Figure 14. 904EXP Airfoil Grid – Concave Region

4.7 ICE_GRID_2D Output

The output from the ICE_GRID_2D module consists of a file containing the grid and informational messages to *stdout*. The grid is output to a file in PLOT3D, multizone, unformatted, two-dimensional, grid format [9]. The grid may be viewed using the "display grid" directive from ICEG2D. The output file name is associated with *grid_file* with the default being *case_name.grd*. A sample of the information printed to *stdout* is shown below:

```
Generating grid using:

       Distribution file     - 623exp.dst
       Grid parameter file   - grid_gen.ini
       Block definition file - new_block.ini
       Grid output file      - 623exp.grd


 ***** Grid Parameter Data *****

       itMax            =  20
       nOrder           =  2
       omega            =  1.00
       addedDissipation =  1.00
       iWake            =  40
       distWake         =  15.00


 ***** Block Parameter Data *****

       nBlocks               =  1

       BLOCK#1:
       distFirstPoint LE        =  0.100000E-06
       distFirstPoint TE        =  0.100000E-05
       distFirstPoint DSB       =  0.100000E-05
       xTransLower              =  0.500000E+00
       xTransUpper              =  0.500000E+00
       distOuterBoundary        =  15.00
       jMaxDist                 =  11



 ***** Information for grid: 623exp - Single block, C-type grid

       IMAX =   699, JMAX =    11


       CONVERGENCE HISTORY

       J   IT        RMS     IRMAX      RMAX
       2    1    0.26457E-07  488    0.38100E-06
       2    2    0.95764E-08  488    0.19727E-06
       .
       .
```

.

An echo of the namelist GRID_PARAMETERS and an echo of the namelist BLOCK_PARAMETERS follow an inventory of the relevant associated files. The grid dimension is then followed a grid informational line. The convergence history is output where J is the index of the line being smoothed, IT is the smoothing iteration number for that line, RMS is the RMS error for the line, and IRMAX and RMAX are the location of the maximum error and the maximum error respectively. In the case of a double block grid, the BLOCK_PARAMETER data and convergence history are repeated.

Grid quality checks are made to ensure that a viable grid has been generated and are included in the output. Specifically, the cell areas in each grid are tested to be sure they are positive. If any negative areas are found, execution in the batch mode is terminated. Further, the grid angles are evaluated to check for cell skewness. Currently, no action is taken except to indicate the minimum angle and display, using a list, the percentage of cell angles within certain ranges. Below is output to *stdout* illustrating the grid quality measures:

```
***** Grid Quality Measures *****

***** Cell Areas *****

      All cell areas positive!

***** Orthogonality *****

      Minimum Angle = 47.73

      Angle distribution        %
      75.0 < angle < 90.0 -  90.6458
      60.0 < angle < 75.0 -   9.2721
      45.0 < angle < 60.0 -   0.0822
      30.0 < angle < 45.0 -   0.0000
             angle < 30.0 -   0.0000

Grid Generation complete!
```

5. NPARC Specific Modules

The modules NPARC_RESTART and NPARC_INPUT generate the files necessary to execute nparc2ds. NPARC_INPUT takes as input problem definition information and grid information to generate an NPARC namelist input file with the internal association *case_name.inp.* NPARC_RESTART takes problem definition information and the grid to generate an NPARC restart file with the internal association *case_name.rst*. Once these two files are defined, the ICEG2D directive "generate solution" may be used to start an NPARC solution run as a background process. The user is referred to [1] for information regarding NPARC output, etc. Both modules assume that a C-type grid is used with the positive direction of the streamwise index taken to be clockwise.

Both NPARC_INPUT and NPARC_RESTART are written in FORTRAN90 with C driver routines. ICEG2D passes four file names as command line arguments to NPARC_INPUT: the NPARC information file, the grid file, the file containing the namelist GRID_PARAMETERS, and the output file for the NPARC namelist. Similarly, ICEG2D passes three file names as command line arguments to NPARC_RESTART: the NPARC information file, the grid file, and the restart file. In the sections that follow, the input and output for each module are described.

5.1 Input for NPARC_INPUT and NPARC_RESTART

Common to the input for both modules is the NPARC information file. The NPARC information file is internally associated with *nparc_info_file* and consists of five namelists: TITLES, INPUTS, TURBIN, SEQDT, and BLOCK. These namelists and their variables are identical to those defined in the NPARC User's Manual [1] with the exception of the variable *nclcd* in namelist INPUTS. This variable was added to specify the frequency of the lift, drag, and moment calculation added by Chung [17]. For definitions of the remaining variables, please refer to [1]. The module NPARC_INPUT is executed using the ICEG2D directive "generate input." Similarly, the module NPARC_RESTART is executed using the ICEG2D directive "generate restart."

The file containing the grid is internally associated with *grid_file* and the file containing the namelist GRID_PARAMETERS is internally associated with *grid_parm_file*. The namelist GRID_PARAMETERS is needed because determination of the trailing edge point from the grid file requires the variable *iWake*.

5.2 NPARC_INPUT Output

The output from NPARC_INPUT consists of a file containing five namelists: TITLES, INPUTS, TURBIN, SEQDT, and BLOCK. The namelist INPUTS is modified through the addition of *nclcd* (described previously) and *jtail1*, which corresponds to the index of the trailing edge point on the lower surface, and *jtail2*, which corresponds to the index of

the trailing edge on the upper surface. Both were included as part of the lift, drag, and moment calculation [17]. The default output for NPARC_INPUT is internally associated

Namelist Variable	
title	5 lines of a title

Table 8. Namelist TITLES Variables

Namelist Variable	**Default Value**
xmach	0.3
re	1.0×10^{6}
alpha	0.0
gamma	1.4
p0sd	0.71429
t0sd	1.0
pref	14.7
trefr	500.0
nc	0
nmax	100
dtcap	100.0
pcqmax	10.0
isolve	1
ivardt	2
irealt	0
realdt	5.0×10^{-6}
tstart	0.0
nsprt	10
np	0
iplot	1
ifxplt	0
ifxprt	0
l2plot	0
dis2	0.25
dis4	0.64
ispect	2
ifiltr	1
splend	0.5
imass	1
iaxisys	0
stopl2	1.0×10^{-12}
nskip	2
numdt	1
nclcd	100

Table 9. Namelist INPUTS Variables

Namelist Variable	Default Values
imutur	2
nturb	2000
imutr2	15

Table 10. Namelist TURBIN Variables

Namelist Variable	Default Values (assuming `numdt` = 2)
dtseq(1)	1.0
iterdt(1)	2000
dtseq(2)	0.7
iterdt(2)	68000

Table 11. Namelist SEQDT Variables

Namelist Variable	Default Values
invisc	1
lamin	1

Table 12. Namelist BLOCK Variables

with *case_name.inp* or an alternative name specified using the ICEG2D directive "define input *NPARC_input_file*." Output from NPARC_INPUT also includes information messages printed to *stdout*. Sample output is shown below:

```
Generating NPARC input file using:

NPARC info file     - NPARC_info_local.npc
Grid file           - 623exp.grd
Grid parameter file - grid_gen.ini
NPARC Info file     - NPARC_info_local.npc

Case Name:

# Case Name
#
# May be five lines
#
#

Grid file - 623exp.grd
   Number of blocks =  1
      Block (1): IMAX, JMAX = 699, 101
Lower surface TE -  40
Upper surface TE - 660
```

```
NPARC input file generation complete!
```

The problem title and relevant information from the problem definition file follow the associated file inventory. Finally, grid information is followed by a message indicating successful generation of the input file.

5.3 NPARC_RESTART Output

The output from NPARC_RESTART is an NPARC restart file (please refer to [1] for the specific format). The default internal file association is *case_name.rst* or an alternative name specified using the ICEG2D directive "define restart *NPARC_restart_file*." Output from NPARC_RESTART also includes information messages printed to *stdout*. Sample output is shown below:

```
Generating NPARC restart file using:

NPARC info file - NPARC_info_local.npc
Grid file       - 623exp.grd
Restart file    - 623exp.rst

NPARC Info file - NPARC_info_local.npc

Case Name:

# Case Name
#
# May be five lines
#
#

Mach number     =  0.28
Reynolds number =  0.60000E+07
Angle of attack =  3.00

Grid file - 623exp.grd
   Number of blocks =  1
      Block (1): IMAX, JMAX = 699, 101

NPARC restart file generation complete!
```

As usual, an inventory of the relevant file associations is printed first. This is followed by the title of the problem and information about the grid.

6. Grid Quality Assessment

While simple geometrical checks can be made to ensure that all cell areas are positive, excessive skewness does not occur, etc., the usefulness of a grid for a specific calculation can only be assessed by determining the accuracy of results predicted using the grid. In this case, computations are made using NPARC for iced airfoil cases for which experimental data exists [8]. It is well understood and appreciated that many factors other than grid quality influence the numerical solution. It is not the intention here to perform an exhaustive validation of the NPARC code and its various turbulence models but only to demonstrate that quality calculations can be made using the grids automatically generated using ICEG2D.

6.1 Comparison of Computed Results with Experimental Data

The available experimental data is for the NLF0441 airfoil section. The data were obtained for the clean airfoil shape and several iced airfoil shapes. The Mach number for each case was approximately 0.3 and the Reynolds number was approximately 6.4×10^6. The range of angles of attack varied for each configuration but generally ranged from –4 degrees to just beyond stall. Force and moment data were obtained along with pressure coefficient data [8]. In all cases below, the numerical solution was generated using the Mach and Reynolds numbers reported for each case. Additionally, the corrected angle of attack, as determined from experimental data, was used. Default values of the various parameters were used except where noted.

6.1.1 The Clean Airfoil

The first case considered here is clean airfoil. Figure 15 shows a region near the airfoil of the single-block, 329x101 grid. As expected, points are clustered near the leading edge of the airfoil due to the curvature. Figure 16 shows a comparison plot of the pressure coefficient on the airfoil surface predicted using NPARC and the experimental data for an angle of attack of 12 degrees. This case corresponds to a condition near the stall condition. The predicted pressure distribution shows very good agreement with the experimental data. It should be noted that the apparent level of agreement is a little misleading in that the differences are larger than they appear. The magnitude of the upper surface suction peak helps mask these differences.

Figure 17 shows a plot of the computed and experimental section lift coefficient curves. In both cases, the lift was determined by integrating the pressures on the surface of the airfoil. The agreement between the two data sets is very good. The NPARC lift curve shows excellent agreement with experimental data near the onset of stall. Further, the predicted post-stall lift coefficient also shows good agreement with experimental data. Finally, Figure 18 shows a plot of the computed and experimental drag curves versus angle of attack. As with the lift coefficient, the drag coefficient was computed by integrating the surface pressures. The agreement is again very good.

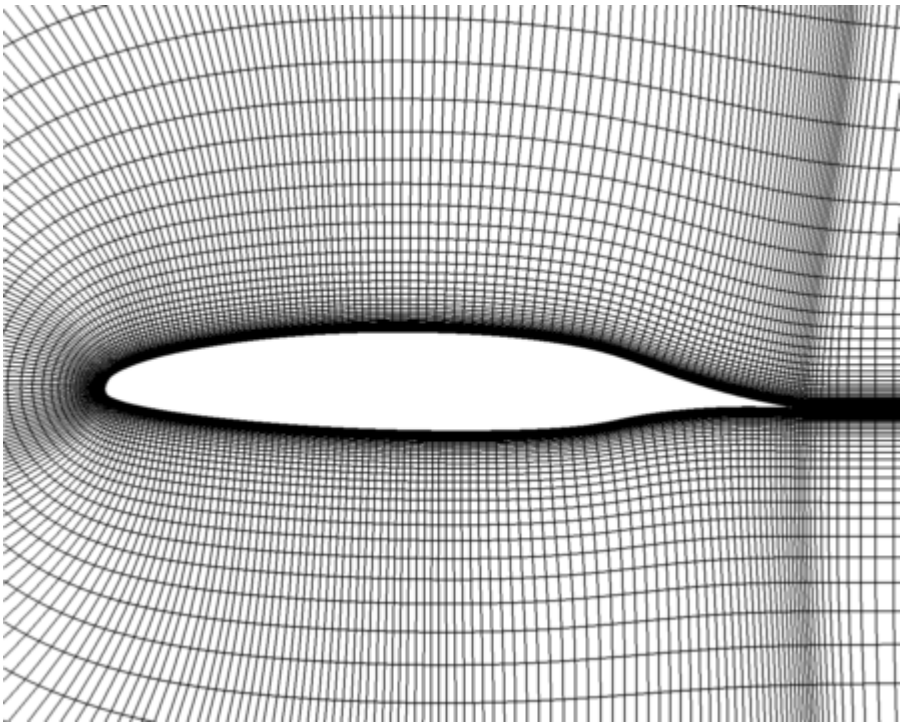


Figure 15. NLF0441 Clean Airfoil – Grid Dimensions 329x101

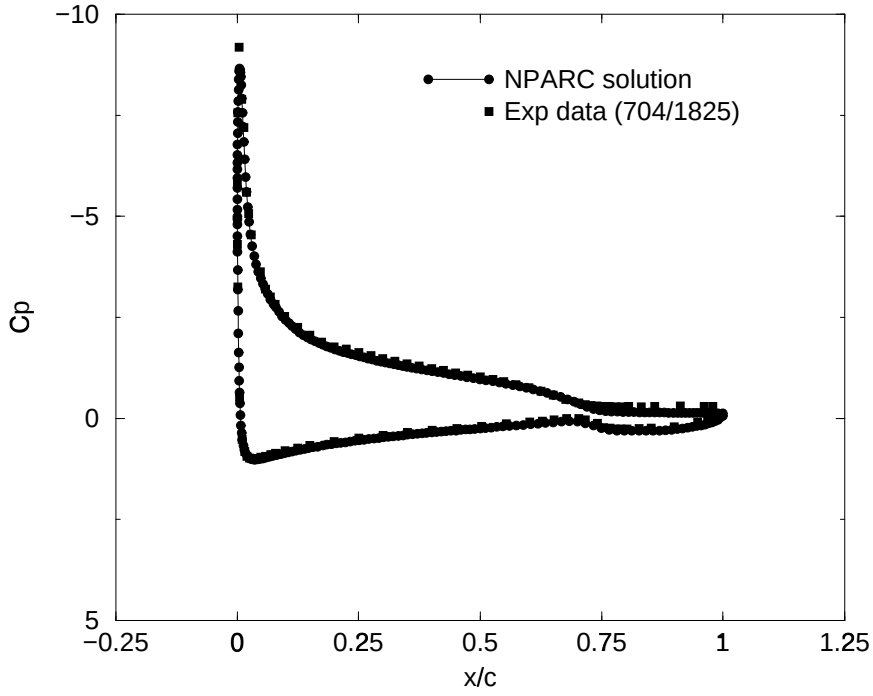


Figure 16. NLF0441 Pressure Distribution – Angle of Attack 12 deg.

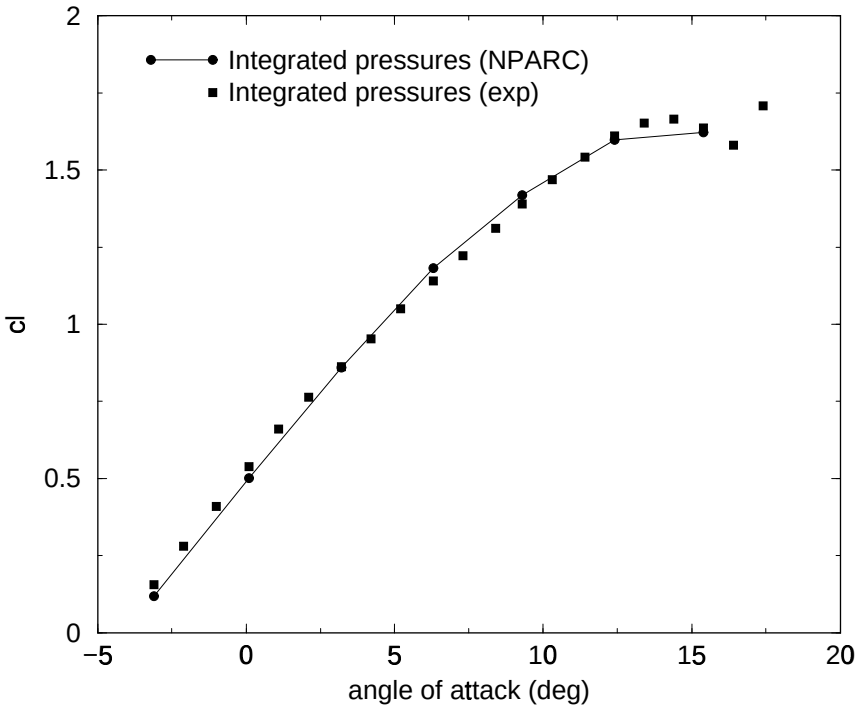


Figure 17. NLF0441 Section Lift Coefficient versus Angle of Attack

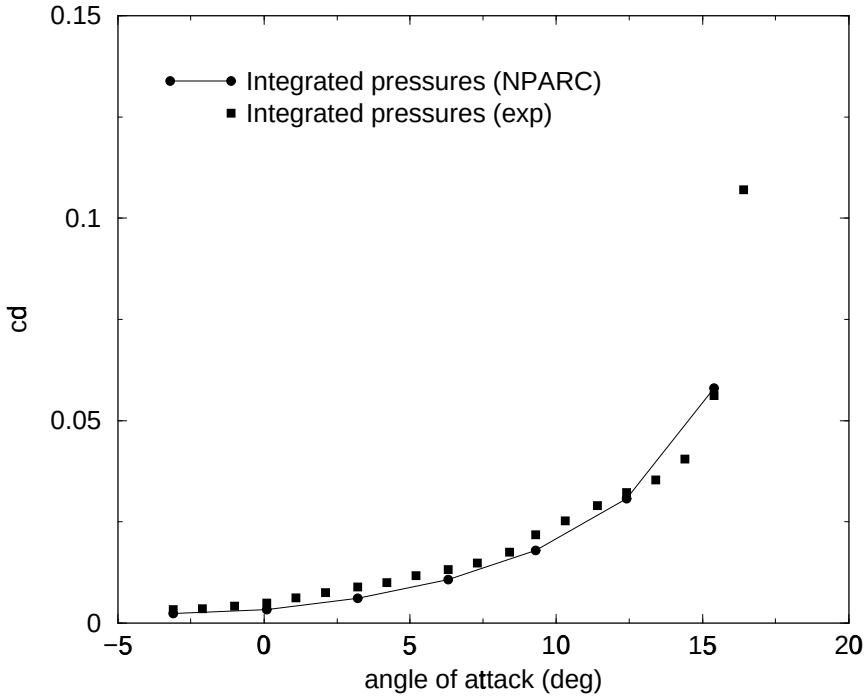


Figure 18. NLF0441 Section Drag Coefficient versus Angle of Attack

6.1.2 Case 622EXP

Having established the fact that the grid distribution algorithm is effective for a clean airfoil, we now consider an iced airfoil case. Case 622EXP corresponds to an NLF0441 airfoil with a "small" ice accretion near the leading edge. The near-airfoil region of the single-block, 529x101 grid automatically generated using ICEG2D is shown in Figure 19. In this case, the "def surface rough" directive was used. Figure 20 shows the leading-edge region of the grid for this case. Figure 21 shows a comparison of predicted and experimental lift coefficients for several angles of attack. Figure 22 shows a comparison of predicted and experimental drag coefficients for several angles of attack. Good agreement between predicted and experimental lift coefficients is exhibited at the lower angles of attack. However, there is significant deviation near stall. The predicted drag coefficient does not share the same level of agreement. Although the overall shapes of the two drag curves are similar, the numerical simulation significantly overestimates the pressure drag at lower angles of attack and underestimates its value beyond stall.

Figure 23 shows a comparison of the predicted and experimental pressure distributions for case 622EXP for an angle of attack of six degrees. At first glance, the comparison appears good. However, upon closer inspection, it is clear that there are significant differences between the predicted and experimental pressure distribution at the leading edge on the upper surface. Figure 24 shows this comparison for the first 10 percent of the chord. Figure 24 shows that, while the agreement between the predicted and experimental data on the lower surface is quite good, relatively poor agreement occurs on the upper surface. While the effect on the lift coefficient was relatively small (see Figure 21), it is clear that the separation region just downstream of the ice accretion is not being predicted accurately. This conclusion is further reinforced by the results shown in Figure 24. Figure 25 shows a comparison of predicted and experimental pressure surface pressure distributions for case 622EXP at an angle of attack of nine degrees. Again, very good agreement is obtained between the predicted and experimental pressure distribution on the lower surface. However, poor agreement is obtained for the upper surface where the NPARC solution is completely separated. This result is consistent with the behavior observed in the lift coefficient (see Figure 21) where stall is predicted at a lower angle of attack than was observed in the experimental data.

Based on the apparent quality of the grid shown in Figure 20, it seems unlikely that the grid is responsible for the lack of agreement between the experimental and predicted results. Additionally, the predicted pressure distribution on the lower surface shows good agreement with the experimental data. These two observations suggest investigating the effects of using a different turbulence model. The baseline model used here is the Spalart-Allmaras [21] one-equation model. NPARC has a suite of turbulence models implemented in the code [1]. To investigate the influence of the turbulence model, calculations were made on the same grid using the two-equation κ–ε model of Chien [22]. Although, these results were quantitatively different than the results obtained using the Spalart-Allmaras model, they did not represent a significant improvement and are not shown here.

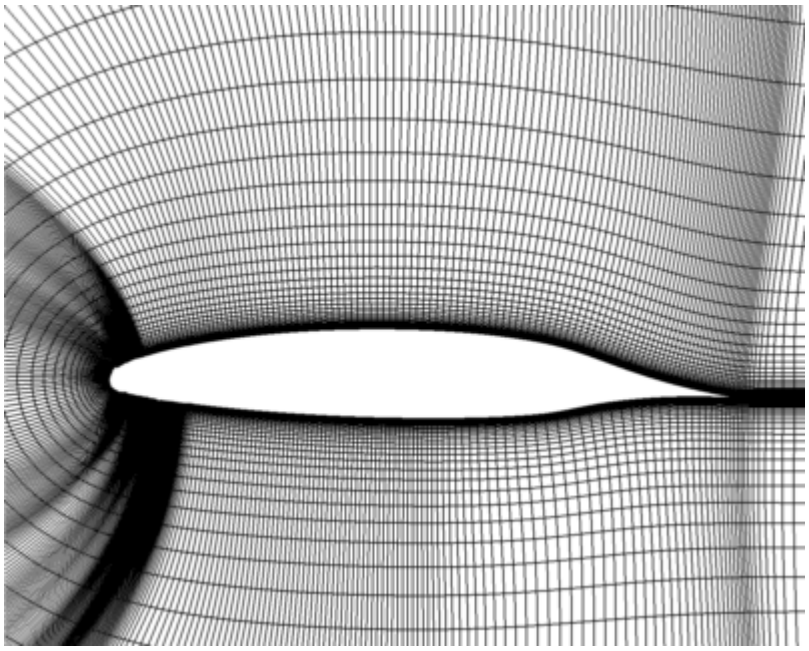


Figure 19. 622EXP – Grid Dimensions 529x101

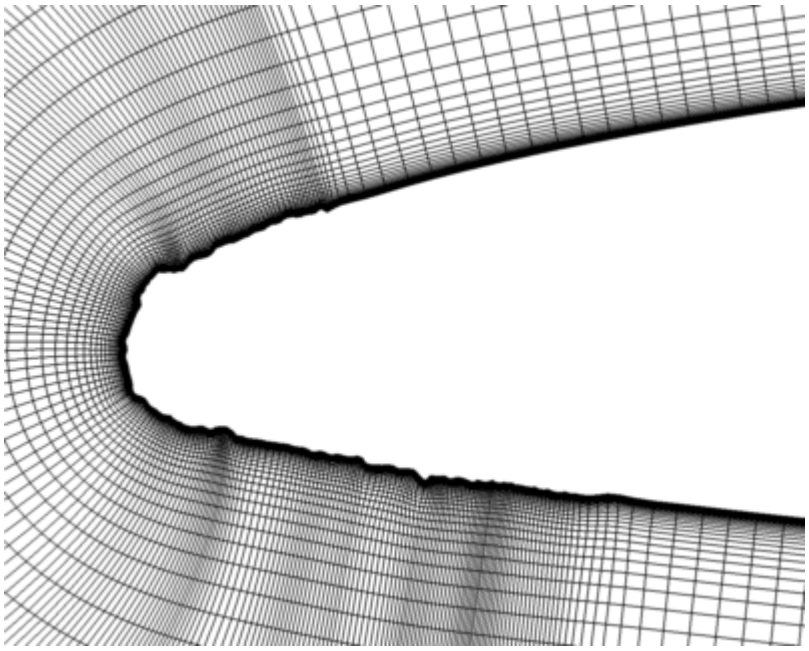


Figure 20. 622EXP Grid in Region Near Nose

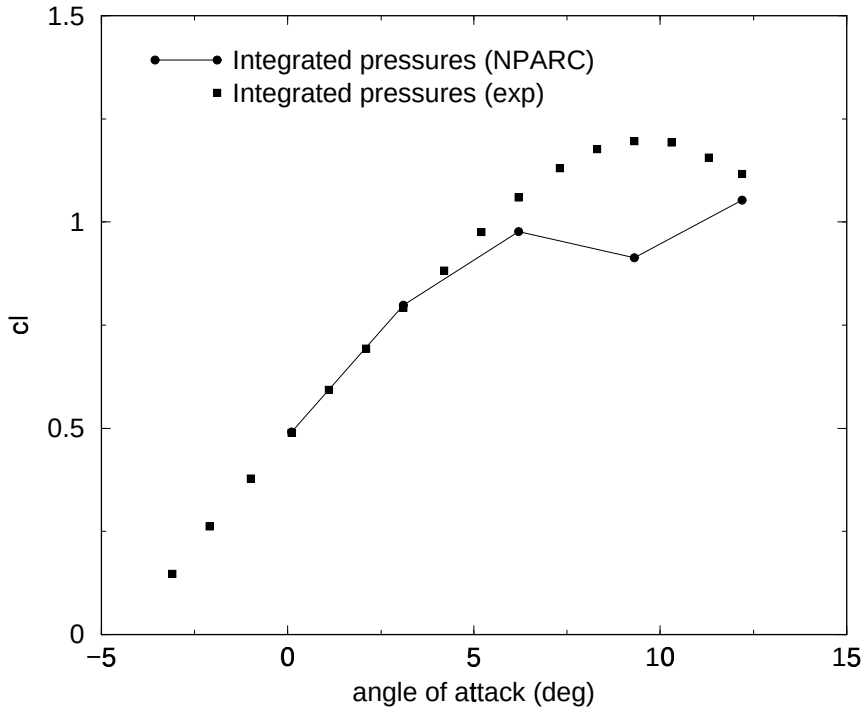


Figure 21. 622EXP Section Lift Coefficient versus Angle of Attack

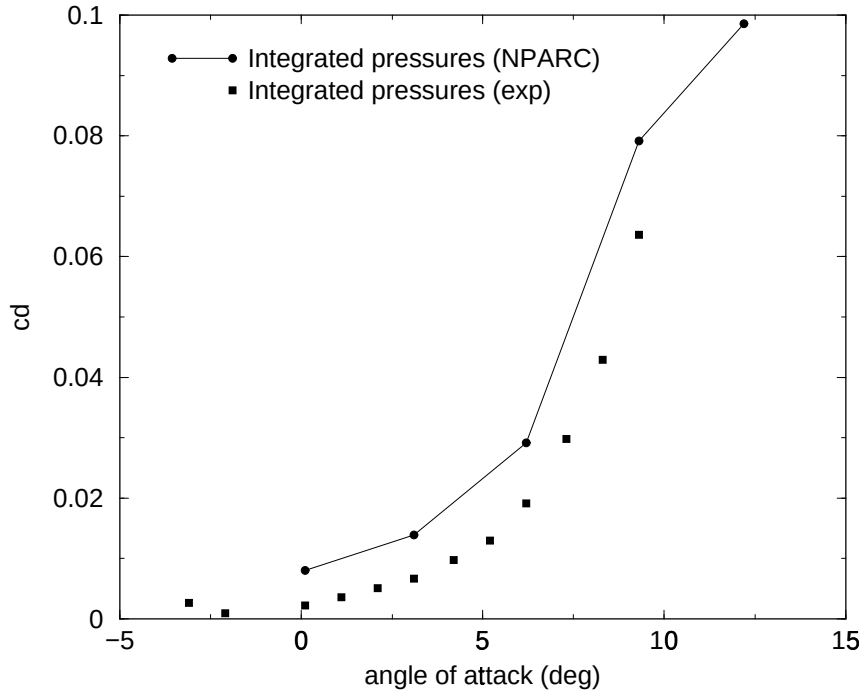


Figure 22. 622EXP Section Drag Coefficient versus Angle of Attack

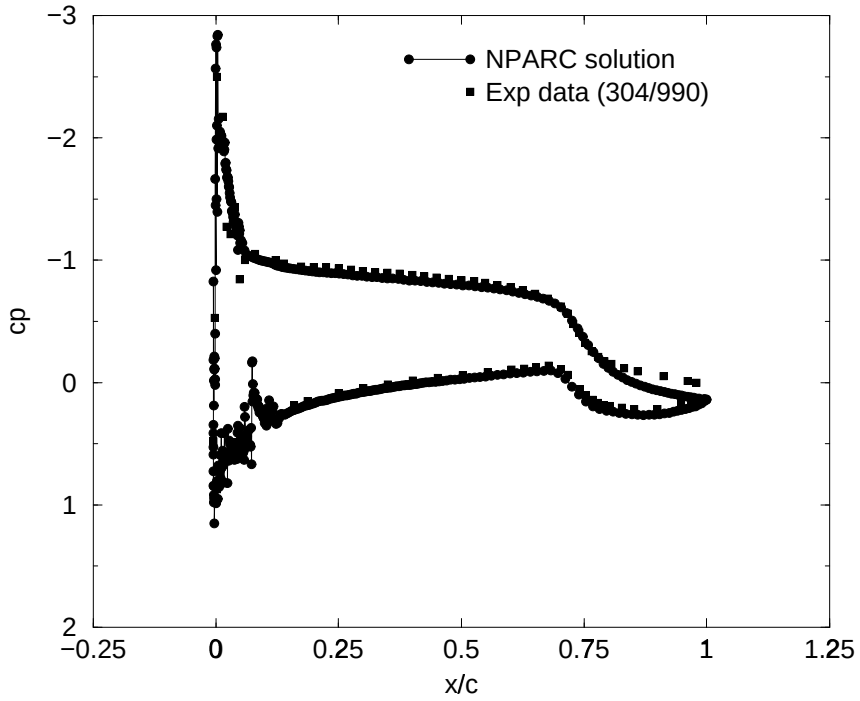


Figure 23. 622EXP Pressure Distribution – Angle of Attack 3 deg.

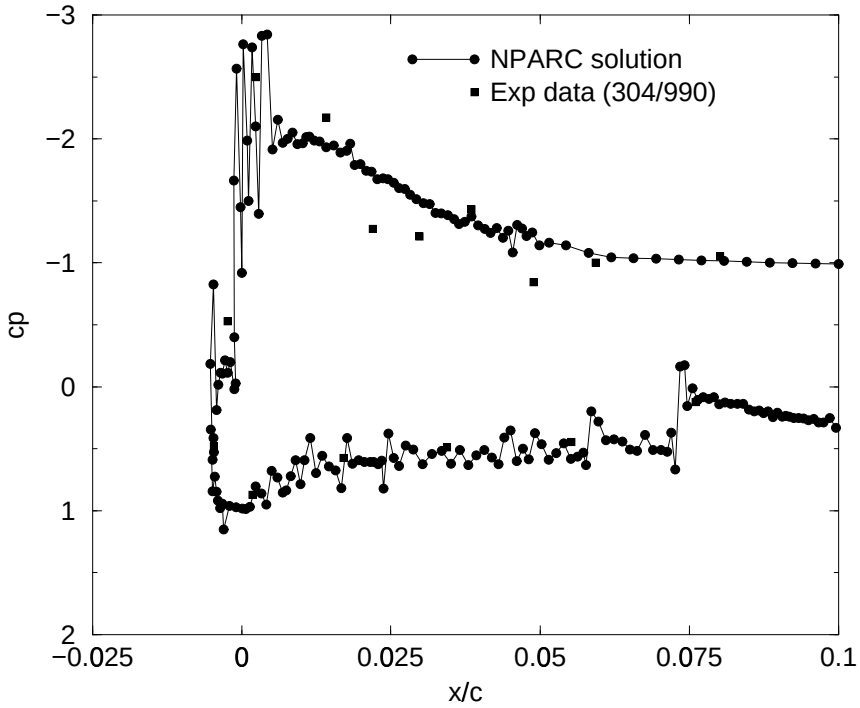


Figure 24. 622EXP Pressure Distribution – Angle of Attack 3 deg.

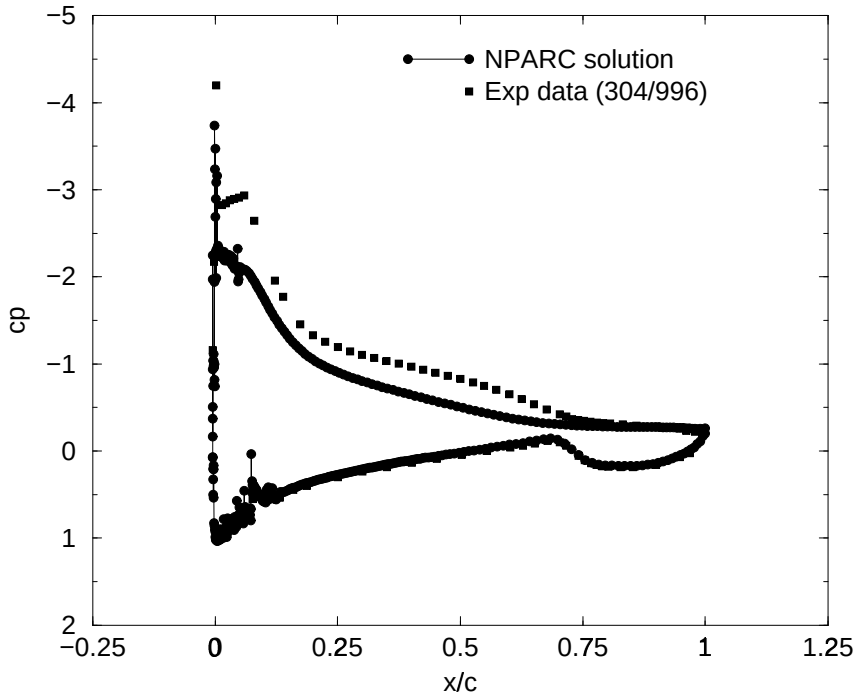


Figure 25. 622EXP Pressure Distribution – Angle of Attack 9 deg.

6.1.3 Case 623EXP

We now consider a case with a significant horn accretion on the leading edge of the airfoil. Figure 26 shows the near-airfoil region in a 699x101 single-block grid automatically generated for the ice shape defined in Case 623EXP. The “def surface rough” directive was used. This case is particularly challenging because of the ice growths on the lower surface of the airfoil that effectively remove points from the horn region. Figure 27 shows the region of the grid near the nose of the airfoil. Figure 28 shows a comparison between section lift coefficients obtained from the NPARC solution and from experimental data for various angles of attack. It should be noted that the computed lift values at the higher angles of attack are actually averages of unsteady lift values. No steady-state solutions were obtained for the six and nine degree angle of attack cases. Figure 29 shows a comparison of the computed and experimentally determined surface pressure distribution for an angle of attack of three degrees. Although the lift value for this case shows only about a 15 percent error, the upper surface pressure distribution shows that the simulation does not do a good job of modeling the flow on the upper surface. In particular, the region just downstream of the horn exhibits poor agreement with the experimental data. No other turbulence models were tested for this case.

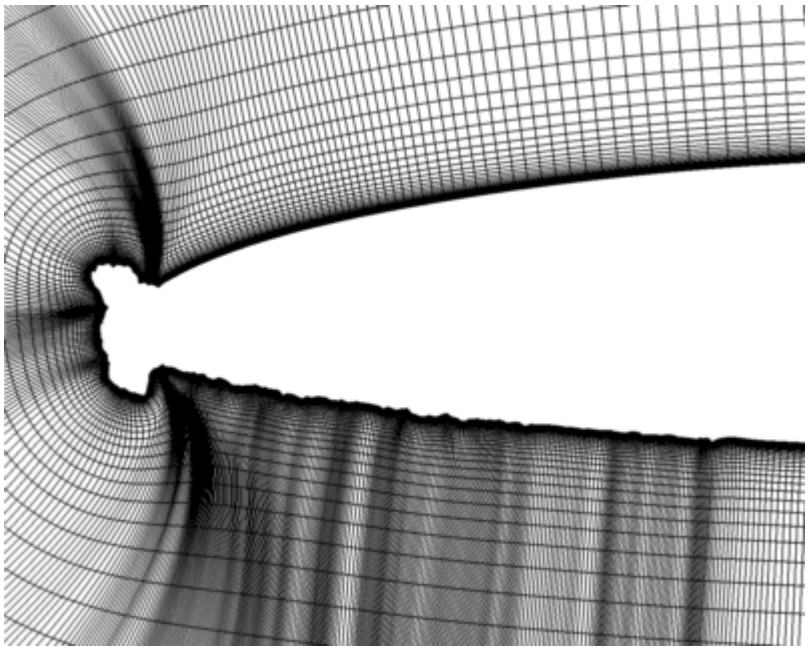


Figure 26. 623EXP – Grid Dimensions 699x101

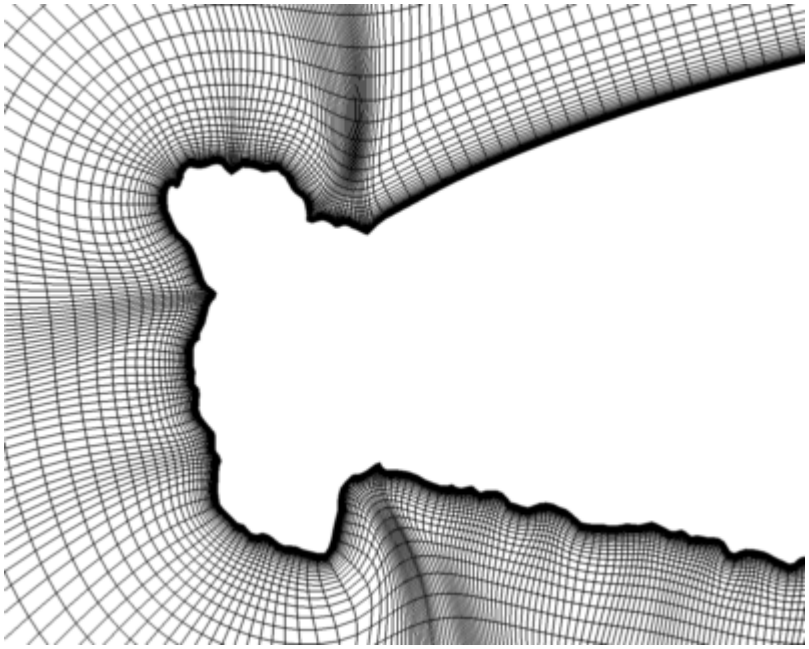


Figure 27. 623EXP Grid in Region Near Nose

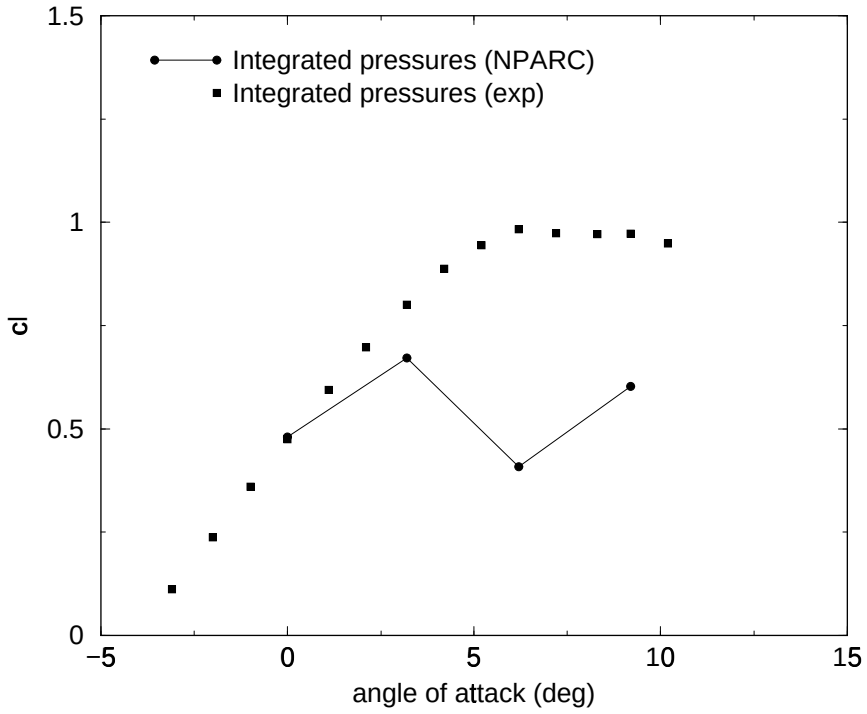


Figure 28. 623EXP Section Lift Coefficient versus Angle of Attack

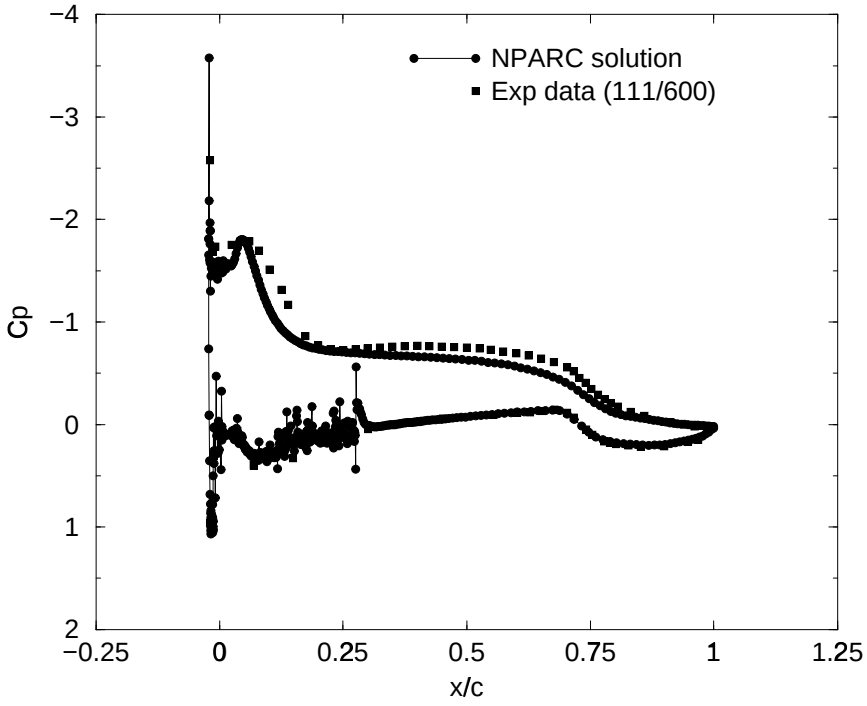


Figure 29. 623EXP Pressure Distribution – Angle of Attack 3 deg.

6.2 Validation of ICEG2D

Validation of the ICEG2D system requires that each component of the system be thoroughly checked-out acting along and in concert with the other modules. This procedure was followed for each module along with the ICEG2D framework and user interface.

The automatic surface definition technology implemented in ICEG2D does a good job of faithfully representing the iced airfoil geometry. The approach implemented here can be thought of as a geometric adaptive grid algorithm using point redistribution. That is, the surface grid distribution is based on geometric features. The feature detection algorithm is based on surface curvature and more points are automatically placed in regions of high curvature at the expense of regions with little or no curvature. We believe this to be one of the strengths of the software package. Since the sole criterion for evaluating the quality of the surface distribution is the fidelity of the geometric representation, we can state unequivocally that the surface distribution algorithm has been validated. Having stated this, however, there are two related questions dealing with solution quality that remain to be answered:

1. How accurately must the surface be represented for CFD calculations?
2. How many points are needed to adequately define the iced airfoil surface?

These questions can only be answered by performing sensitivity analyses that are the subject of further research.

Validation of the grid generation algorithm is partially achieved using the error checks built into ICEG2D, i.e., insuring that all cell areas are positive and checking the skewness of the cells. In this sense, the grid generation module has been thoroughly validated. Also implied in the validation of a grid generation algorithm is an evaluation of results predicted using the grids generated by the algorithm. Comparison of numerically predicted results with experimental data indicates that the grid generation algorithm is effective for cases where there is minimal flow separation. Here we define effective to mean that the agreement between predicted results and experimental data is good. The agreement between numerical results and experimental data is degraded when significant flow separation occurs. However, there are many factors that influence the accuracy of a numerical solution beyond the grid. An attempt was made to determine the influence of the selected turbulence model. As stated previously, no significant improvement was observed using the two-equation model of Chien. At this point, we can make no strong conclusions regarding the efficacy of the grid generation algorithm for these cases without further research.

7. Examples

Included in this section are several example ICEG2D directive streams illustrating the use of ICEG2D. These input streams and the data needed to run these cases are included in the ICEG2D distribution in the directory $ICEG2D_DIR/examples.

The first case considered is one in which we want to generate a single-block grid for an experimental ice shape. The data for this case is contained in the directory $ICEG2D_DIR/examples/example_1. We assume the ice shape to be smooth. The experimental ice shape is defined in the ASCII file **newdata.dat** and the clean airfoil shape is defined in the ASCII file **newdata.cln**. In this instance, the file extensions are appropriate for the data types for a case name of `newdata`. There is also an NPARC information file named **newdata.npc** in the local directory. The ICEG2D directive stream for an interactive session is listed below:

```
ICEG2D> def case newdata
ICEG2D> gen dist
ICEG2D> gen grid
ICEG2D> display grid
ICEG2D> def nparc newdata.npc
ICEG2D> gen input
ICEG2D> gen restart
ICEG2D> exit
```

In this case, the default parameter file associations are used. Also, note that the minimal versions of the directives are used, i.e., “def” instead of “define.” The restart file is named **newdata.rst**, the input file is named **newdata.inp**, etc.

Now consider a case that we want to execute in batch mode with an experimental ice shape defined in the ASCII file **904iced.af** and a clean airfoil defined in the file **glc305.af**. These data files and the ASCII driver file **driver.inp** are located in the directory $ICEG2D_DIR/examples/example_2. The NPARC information file is **input.npc**. We want to use a modified DISTRIBUTION namelist in the file **dist_local.ini**. We also want to generate a double-block grid using the default double-block parameter file **double_block.ini**. The directive stream contained in **driver.inp** is listed below:

```
def mode batch
def case 904iced
def iced_geom 904iced.af
def clean_geom glc305.af
def dist_parm dist_local.ini
gen dist
def block_parm double
gen grid
def nparc input.npc
gen input
gen restart
# gen solution
exit
```

The ICEG2D script is executed using "`ICEG2D < driver.inp > output`." In this case, the input file is named **904iced.inp**, etc. If user-defined blocking parameters are to be used and are defined in the local file **block_def.inp**, the directive "def block_parm double" should be replaced with "def block_parm block_def.inp". Note that the "gen solution" directive has been commented out.

We now consider a case starting from an intermediate step. Here, we assume a single-block grid file exists that is in the PLOT3D unformatted, two-dimensional, multizone, grid format [9] and is named **grid.xyz**. The NPARC information file is located in **prob.inp**. Both files are located in the directory $ICEG2D_DIR/examples/example_3. The directive stream for an interactive session is listed below:

```
ICEG2D> def case problem1
ICEG2D> def grid grid.xyz
ICEG2D> def nparc prob.inp
ICEG2D> show
ICEG2D> gen input
ICEG2D> gen restart
ICEG2D> gen solution
ICEG2D> exit
```

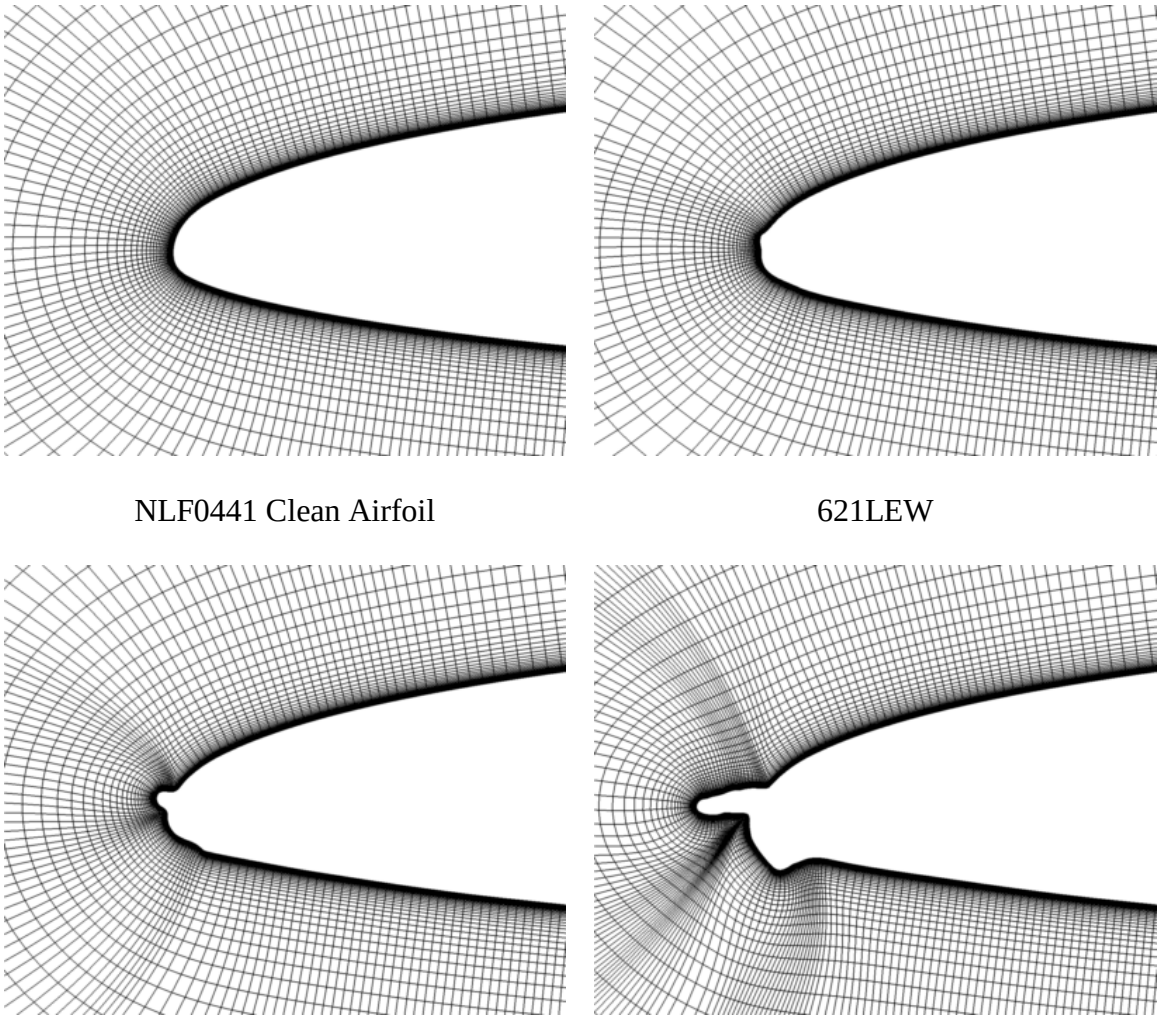
Although the file associations required by the modules ICE_DIST_2D and ICE_GRID_2D are not needed, we still need a case name for the default file associations for the NPARC input file, the restart file, etc. Note that the "show" directive lists all the current file associations and is useful for checking to make sure everything has been defined as intended.

Finally, a directive stream is included to illustrate how ICEG2D could be used to automatically generate grids for an icing calculation. The files for this case are contained in $ICEG2D_DIR/examples/example_4. The data represent a sequence of iced airfoil geometries starting from the clean airfoil **nlf0441.cln**, proceeding through **621lew.dat**, **622lew.dat**, and concluding with **623lew.dat**. The directive stream to simulate automatic regridding is given by:

```
#
# Script for auto-gridding for NLF0441
#
# Set mode to batch
#
def mode batch
#
# Use smooth surface parameters for each case (the default)
#
# Clean airfoil
#
def surface smooth
def case nlf0441
def iced_geom nlf0441.cln
gen dist
#
# Use the default single block topology
```

```
#
def block_parm single
gen grid
#
# Case 621lew
#
def surface smooth
def case 621lew
def clean_geom nlf0441.cln
gen dist
#
def block_parm single
gen grid
#
# Case 622lew
#
def surface smooth
def case 622lew
def clean_geom nlf0441.cln
gen dist
#
def block_parm single
gen grid
#
# Case 623lew
#
def surface smooth
def case 623lew
def clean_geom nlf0441.cln
gen dist
#
def block_parm single
gen grid
#
exit
```

Notice that after each new “def case” directive, the clean airfoil file must be respecified. This is because it is automatically redefined by the “def case” directive. Of course, had this been an actual regridding calculation, calls to the ice growth prediction code LEWICE as well as the LEWICE input modules would need to be included. Figure 30 shows the sequence of grids automatically generated using ICEG2D for the simulated ice growth.



NLF0441 Clean Airfoil

621LEW

622LEW

623LEW

Figure 30. Automatic Regridding for Simulated Ice Growth

8. Summary

An integrated geometry/grid/simulation software package, ICEG2D, was developed to automate computational fluid dynamics (CFD) simulations for single-element airfoils with ice accretion. ICEG2D was designed to automatically perform three primary functions: 1) generate a grid-ready, surface definition based on the geometrical characteristics of the iced airfoil surface, 2) generate a high-quality grid using the generated surface point distribution, and 3) generate the input and restart files needed to run the general purpose CFD solver NPARC. ICEG2D can be operated in either a batch mode using a script file or in an interactive mode by entering directives from a command line from within a Unix shell. ICEG2D consists of four modules that are integrated with a framework of associations that relate external data to internal file names.

ICEG2D has been tested by application to numerous iced airfoil configurations. The geometry module of ICEG2D was validated based on its ability to distribute points on the surface of the iced airfoil while maintaining geometric fidelity. Additionally, the geometry module has the ability to merge incomplete iced airfoil descriptions with clean airfoil shapes and to prune ice shapes with loops and twists. The grid generation module was validated based on two criteria. The first, a necessary condition, is based on geometric characteristics of the criteria, i.e., cell area positivity and no excessive skewness. The second condition is based on the accuracy of solutions computed using the automatically generated grids. The first condition was satisfied. At this time, the validity of the grids for problems with significant regions of flow separation has not been demonstrated and is the subject of further research.

9. References

1. NPARC User's Guide Version 3.0, 1996.
2. G. B. Hong, "GGLib: A Geometry & Grid Generation Library," ERC Technical Report, Mississippi State University, May 1999.
3. S. Nakamura, "Noniterative Grid Generation Using Parabolic Partial Differential Equations", Numerical Grid Generation, Ed. J. F. Thompson, Elsevier Science Publishing Company, Inc., New York, 1982.
4. R. W. Noack, and D. A. Anderson, "Solution-Adaptive Grid Generation Using Parabolic Partial Differential Equations", AIAA J., 28, 1016-1023 (1990).
5. R. W. Noack, and I. H. Parpia, "Solution Adaptive Parabolic Grid Generation in Two and Three Dimensions", Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Eds. A. S.-Arcilla, J. Hauser, P. R. Eiseman, and J. F. Thompson, Elsevier Science Publishing Company, New York, 1991.
6. D. S. Thompson and B. K. Soni, "Generation of Quad- and Hex-Dominant, Semistructured Meshes using an Advancing Layer Scheme," Proc. 8th International Meshing Roundtable, 171-178, October 1999.
7. D. S. Thompson and B. K. Soni, "Semistructured Grid Generation in Three Dimensions using a Parabolic Marching Scheme," AIAA Paper 2000-1004, Presented at the 38th Aerospace Sciences Meeting, Reno, NV, January 2000.
8. J. K. Chung, private communication, August 1999.
9. FAST User Guide. <http://www.nas.nasa.gov/Software/FAST/>, June, 1999.
10. W. B. Wright, "A Summary of Validation Results for LEWICE 2.0," AIAA Paper 99-0249, Presented at the 37th Aerospace Sciences Meeting, Reno, NV, January 1999.
11. H. E. Addy, M. G. Potapczuk, and D. W. Sheldon, "Modern Airfoil Ice Accretions," NASA TM 107423, March 1997.
12. J. Chung, Y. Choo, A. Reehorst, M. Potapczuk, and J. Slater, "Numerical Study of Aircraft Controllability Under Certain Icing Conditions," AIAA Paper 99-0375, Presented at the 37th Aerospace Sciences Meeting, Reno, NV, January 1999.
13. Y. Choo, J. Slater, T. Henderson, C. Bidwell, D. Braun, and J. Chung, "User Manual for Beta Version of Turbo-GRD -- A Software System for Interactive Two-Dimensional Boundary/Field Grid Generation, Modification, and Refinement," NASA TM-1998-206631, Oct., 1988
14. L. Piegl and W. Tiller, The NURBS Book, 2nd Ed., Monographs in Visual Communication, Springer, 1997.
15. J. K. Chung, private communication, May, 1999.
16. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, Numerical Grid Generation, Elsevier Science Publishing Company, Inc., New York, 1985.
17. J. K. Chung, private communication, June 1999.
18. W. M. Chan and J. L. Steger, "Enhancements of a Three-Dimensional Grid Generation Scheme", App. Math. Comp., 51, 181-205 (1992).
19. G. O. Roberts, "Computational Meshes for Boundary Layer Problems," Proc. Int. Conf. Num. Methods Fluid Dyn., Lecture Notes in Physics, 8, 171-177 (1971).
20. B. K. Soni, "Elliptic Mesh Generation Systems: Control Functions Revisited," App. Math. Comp., 59, 151-164 (1993).

21. P. R. Spalart and S. R. Allmaras, " A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, Presented at the 30th Aerospace Sciences Meeting, Reno, NV, January 1992.
22. K.-Y. Chien, "Predictions of Channel and Boundary-Layer Flows with a Low-Reynolds-Number Turbulence Model, AIAA J., 20, 33-38 (1982)